Deutsch

Fujitsu Software BS2000

openUTM V7.0

Anwendungen administrieren

Benutzerhandbuch

Stand der Beschreibung:
openUTM V7.0A00

Ausgabe November 2019

Kritik… Anregungen… Korrekturen…

Die Redaktion ist interessiert an Ihren Kommentaren zu diesem Handbuch. Ihre Rückmeldungen helfen uns, die Dokumentation zu optimieren und auf Ihre Wünsche und Bedürfnisse abzustimmen.

Sie können uns Ihre Kommentare per E-Mail an bs2000.info@fujitsu.com senden.

Zertifizierte Dokumentation nach DIN EN ISO 9001:2015

Um eine gleichbleibend hohe Qualität und Anwenderfreundlichkeit zu gewährleisten, wurde diese Dokumentation nach den Vorgaben eines Qualitätsmanagementsystems erstellt, welches die Forderungen der DIN EN ISO 9001: 2015 erfüllt.

Copyright und Handelsmarken

Copyright © 2025 Fujitsu

Alle Rechte vorbehalten.
Liefermöglichkeiten und technische Änderungen vorbehalten.

Alle verwendeten Hard- und Softwarenamen sind Handelsnamen und/oder Warenzeichen der jeweiligen Hersteller.

Inhaltsverzeichnis

Anwendungen administrieren 11
1 Einleitung 12
1.1 Zielgruppe und Konzept des Handbuchs 14
1.2 Wegweiser durch die Dokumentation zu openUTM 15
1.2.1 openUTM-Dokumentation 16
1.2.2 Dokumentation zum openSEAS-Produktumfeld 19
1.2.3 Readme-Dateien 20
1.3 Änderungen in openUTM V7.0 21
1.3.1 Neue Server-Funktionen 22
1.3.2 Entfallene Server-Funktionen 26
1.3.3 Neue Client-Funktionen 27
1.3.4 Neue Funktionen für openUTM WinAdmin 28
1.3.5 Neue Funktionen für openUTM WebAdmin 29
1.4 Darstellungsmittel 30
2 Überblick über die openUTM-Administration 32
2.1 Kommandoschnittstelle 34
2.2 Programmschnittstelle KDCADMI 36
2.3 Beispielprogramme 40
2.4 PADM, DADM zur Administration von Message Queues und Druckern 41
2.5 Das Tool CALLUTM (BS2000-Systeme) 42
2.6 openUTM WinAdmin und openUTM WebAdmin 43
3 Objekte administrieren, Parameter einstellen 44
3.1 Informationsfunktionen von openUTM 45
3.2 Performance-Kontrolle 47
3.2.1 Informationen über die Auslastung der Anwendung 48
3.2.2 Diagnose von Fehlern und Engpässen 49
3.2.3 Mögliche Maßnahmen 50
3.3 Pagepool-Engpass vermeiden 56
3.3.1 Pagepool einer stand-alone-Anwendung 57
3.3.2 Pagepools einer UTM-Cluster-Anwendung 60
3.4 Anwendungsprogramm austauschen 61
3.5 Clients und Drucker 62
4 Konfiguration dynamisch ändern 65
4.1 Anforderungen an die KDCDEF-Generierung 66
4.2 Objekte dynamisch in die Konfiguration eintragen 70
4.2.1 Clients, Drucker und LTERM-Partner eintragen 71

4.2.2 Teilprogramme, Transaktionscodes, TAC-Queues und VORGANG-Exits eintragen 74
4.2.3 Benutzerkennungen eintragen 76
4.2.4 Keysets erzeugen 77
4.2.5 LU6.1 - Verbindungen für verteilte Verarbeitung eintragen 78
4.2.6 LTACs eintragen 79
4.2.7 Format und Eindeutigkeit der Objektnamen 80
4.3 Objekte dynamisch aus der Konfiguration löschen 82
4.3.1 Clients/Drucker und LTERM-Partner löschen 84
4.3.2 Teilprogramme, Transaktionscodes und VORGANG-Exits löschen 86
4.3.3 Benutzerkennungen löschen 88
4.3.4 Keysets löschen 90
4.3.5 LU6.1 - Verbindungen und Sessions löschen 91
4.3.6 LTACs löschen 92
4.4 Objekteigenschaften ändern 93
4.4.1 Clients/Drucker und LTERM-Partner modifizieren 94
4.4.2 Transaktionscodes und TAC-Queues modifizieren 95
4.4.3 Benutzerkennungen modifizieren 96
4.4.4 Keysets modifizieren 97
4.4.5 LU6.1 - Sessions modifizieren 98
5 Generierungsanweisungen aus der KDCFILE erzeugen 99
5.1 Starten des inversen KDCDEF 101
5.2 Ergebnis des inversen KDCDEF-Laufs 103
5.3 Inverser KDCDEF bei Versionsübergängen 104
5.4 Empfehlungen für die Neugenerierung einer Anwendung 105
6 Administration über Kommandos 107
6.1 Administration im Dialog 108
6.2 Administration über Message Queuing 109
7 Erstellen eigener Administrationsprogramme 112
7.1 Dialog-Administrationsprogramme 113
7.1.1 Mehrere Administrationsaufrufe 114
7.1.2 Mehrschritt-Vorgang 115
7.2 Diagnosemöglichkeiten für die Administrationsschnittstelle 116
8 Zentrale Administration mehrerer Anwendungen 117
8.1 Administration über WinAdmin und WebAdmin 119
8.1.1 Generierung der UTM-Anwendung anpassen 120
8.1.2 WinAdmin und WebAdmin konfigurieren 122
8.2 Konfigurationsmodelle für eigene Administrationsanwendungen 124
8.2.1 Administration über UPIC-Client 125
8.2.2 Administration über Verteilte Verarbeitung 130
8.2.3 Administration über TS-Anwendung 135

8.3 Zentrale Administration über Kommandos 137
8.4 Zentrale Administration über Programme 138
8.4.1 Dezentrale Administrationsprogramme 139
8.4.2 Zentrale Administrationsprogramme 141
9 Administration automatisieren 143
9.1 Steuerung über MSGTAC-Programm 144
9.2 Steuerung über benutzerspezifische Meldungsziele 147
10 Zugriffsrechte und Zugriffsschutz 148
10.1 Konfiguration der Administrator-Verbindung 151
10.2 Administrationsberechtigung erteilen 152
10.3 Administrationskommandos generieren 153
11 Programmschnittstelle zur Administration - KDCADMI 155
11.1 Aufruf der Funktion KDCADMI 156
11.1.1 KDCADMI-Funktionsaufruf 157
11.1.2 Beschreibung der zu versorgenden Datenbereiche 158
11.1.3 Returncodes 171
11.1.4 Versorgung der Datenstrukturfelder bei der Datenübergabe 175
11.2 Operationscodes von KDCADMI 176
11.2.1 KC_CHANGE_APPLICATION - Anwendungsprogramm austauschen 177
11.2.2 KC_CREATE_DUMP - UTM-Dump erzeugen 185
11.2.3 KC_CREATE_OBJECT - Objekte in die Konfiguration eintragen 187
11.2.3.1 obj_type = KC_CON 193
11.2.3.2 obj_type = KC_KSET 195
11.2.3.3 obj_type = KC_LSES 196
11.2.3.4 obj_type = KC_LTAC 197
11.2.3.5 obj_type = KC_LTERM 200
11.2.3.6 obj_type = KC_PROGRAM 205
11.2.3.7 obj_type = KC_PTERM 207
11.2.3.8 obj_type = KC_TAC 215
11.2.3.9 obj_type = KC_USER 223
11.2.3.10 Returncodes 232
11.2.4 KC_CREATE_STATEMENTS - KDCDEF-Steueranweisungen erzeugen (inverser KDCDEF) 253
11.2.5 KC_DELETE_OBJECT - Objekte löschen 263
11.2.6 KC_ENCRYPT - RSA-Schlüsselpaar erzeugen, löschen, auslesen 277
11.2.7 KC_GET_OBJECT - Informationen abfragen 288
11.2.8 KC_LOCK_MGMT - Sperren in UTM-Cluster-Anwendungen aufheben 316
11.2.9 KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern 321
11.2.9.1 obj_type=KC_CLUSTER_NODE 330
11.2.9.2 obj_type=KC_DB_INFO 331

11.2.9.3 obj_type=KC_KSET 332
11.2.9.4 obj_type=KC_LOAD_MODULE 333
11.2.9.5 obj_type=KC_LPAP 336
11.2.9.6 obj_type=KC_LSES 340
11.2.9.7 obj_type=KC_LTAC 342
11.2.9.8 obj_type=KC_LTERM 344
11.2.9.9 obj_type=KC_MUX (BS2000-Systeme) 348
11.2.9.10 obj_type=KC_OSI_CON 351
11.2.9.11 obj_type=KC_OSI_LPAP 352
11.2.9.12 obj_type=KC_PTERM 357
11.2.9.13 obj_type=KC_TAC 362
11.2.9.14 obj_type=KC_TACCLASS 367
11.2.9.15 obj_type=KC_TPOOL 370
11.2.9.16 obj_type=KC_USER 372
11.2.9.17 obj_type = KC_CLUSTER_CURR_PAR 377
11.2.9.18 obj_type=KC_CLUSTER_PAR 378
11.2.9.19 obj_type = KC_CURR_PAR 380
11.2.9.20 obj_type = KC_DIAG_AND_ACCOUNT_PAR 384
11.2.9.21 obj_type = KC_MAX_PAR 396
11.2.9.22 obj_type=KC_TASKS_PAR 399
11.2.9.23 obj_type = KC_TIMER_PAR 401
11.2.9.24 Returncodes 405
11.2.10 KC_ONLINE_IMPORT - Anwendungsdaten online importieren 423
11.2.11 KC_PTC_TA - Transaktion im Zustand PTC zurücksetzen 427
11.2.12 KC_SEND_MESSAGE - Nachricht senden (BS2000-Systeme) 431
11.2.13 KC_SHUTDOWN - Anwendungslauf beenden 435
11.2.14 KC_SPOOLOUT - Verbindungen zu Druckern aufbauen 445
11.2.15 KC_SYSLOG - System-Protokolldatei administrieren 449
11.2.16 KC_UPDATE_IPADDR - IP-Adressen aktualisieren 462
11.2.17 KC_USLOG - Benutzer-Protokolldatei administrieren 469
11.3 Datenstrukturen zur Informationsübergabe 472
11.3.1 Datenstrukturen zur Beschreibung der Objekteigenschaften 474
11.3.1.1 kc_abstract_syntax_str - Abstrakte Syntax für die Kommunikation über OSI TP 475
11.3.1.2 kc_access_point_str - OSI TP-Zugriffspunkt 476
11.3.1.3 kc_application_context_str - Application Context für die Kommunikation über OSI TP 481
11.3.1.4 kc_bcamappl_str - Namen und Adressen der lokalen Anwendung 482
11.3.1.5 kc_character_set_str - Namen von Character Sets (nur auf BS2000-Systemen) 485
11.3.1.6 kc_cluster_node_str - Knoten-Anwendungen einer UTM-Cluster-Anwendung 486

11.3.1.7 kc_con_str - LU6.1-Verbindungen 492
11.3.1.8 kc_db_info_str - Datenbank-Informationen ausgeben 497
11.3.1.9 kc_edit_str - Optionen von EDIT-Profilen (BS2000-Systeme) 499
11.3.1.10 kc_gssb_str - GSSBs der Anwendung 502
11.3.1.11 kc_http_descriptor_str - HTTP-Deskriptoren der Anwendung 503
11.3.1.12 kc_kset_str - Keysets der Anwendung 505
11.3.1.13 kc_load_module_str - Lademodule (BS2000-Systeme) bzw. Shared Objects /DLLs (Unix-, Linux- und Windows-Systeme) 507
11.3.1.14 kc_lpap_str - Eigenschaften von LU6.1-Partner-Anwendungen 510
11.3.1.15 kc_lses_str - LU6.1-Sessions 516
11.3.1.16 kc_ltac_str - Transaktionscodes ferner Services (LTAC) 519
11.3.1.17 kc_lterm_str - LTERM-Partner 524
11.3.1.18 kc_message_module_str - Benutzereigene Meldungsmodule 535
11.3.1.19 kc_mux_str - Multiplexanschlüsse (BS2000-Systeme) 537
11.3.1.20 kc_osi_association_str - Associations zu OSI TP-Partner-Anwendungen . 541
11.3.1.21 kc_osi_con_str - OSI TP-Verbindungen 543
11.3.1.22 kc_osi_lpap_str - Eigenschaften von OSI TP-Partner-Anwendungen . . 550
11.3.1.23 kc_program_str - Teilprogramme und VORGANG-Exits 557
11.3.1.24 kc_ptc_str - Transaktionen im Zustand PTC 560
11.3.1.25 kc_pterm_str - Clients und Drucker 563
11.3.1.26 kc_queue_str - Eigenschaften temporärer Queues 576
11.3.1.27 kc_sfunc_str - Funktionstasten 577
11.3.1.28 kc_subnet_str - Information zu Subnetzen 579
11.3.1.29 kc_tac_str - Transaktionscodes lokaler Services 580
11.3.1.30 kc_tacclass_str - TAC-Klassen der Anwendung 591
11.3.1.31 kc_tpool_str - LTERM-Pools der Anwendung 594
11.3.1.32 kc_transfer_syntax_str - Transfersyntax für die Kommunikation über OSI TP 603
11.3.1.33 kc_user_str, kc_user_fix_str, kc_user_dyn1_str bzw. kc_user_dyn2_str - Benutzerkennungen 604
11.3.2 Datenstrukturen zur Beschreibung der Anwendungsparameter 623
11.3.2.1 kc_cluster_curr_par_str - Statistikwerte einer UTM-Cluster-Anwendung 624
11.3.2.2 kc_cluster_par_str - Globale Eigenschaften einer UTM-Cluster-Anwendung 625
11.3.2.3 kc_curr_par_str - Aktuelle Werte der Anwendungsparameter 634
11.3.2.4 kc_diag_and_account_par_str - Diagnose- und Accounting-Parameter . 646
11.3.2.5 kc_dyn_par_str - Dynamisch erzeugbare Objekte 654
11.3.2.6 kc_max_par_str - Maximalwerte der Anwendung (MAX-Parameter) . . . 660
11.3.2.7 kc_msg_dest_par_str - Eigenschaften der Benutzer-spezifischen Meldungsziele 678
11.3.2.8 kc_pagepool_str - aktuelle Belegung des Pagepools 680

11.3.2.9 kc_queue_par_str - Eigenschaften von Queue-Objekten 682
11.3.2.10 kc_signon_str - Eigenschaften des Anmeldeverfahrens 683
11.3.2.11 kc_system_par_str - Systemparameter 687
11.3.2.12 kc_tasks_par_str - Anzahl der Prozesse 691
11.3.2.13 kc_timer_par_str - Timer-Einstellungen 695
11.3.2.14 kc_utmd_par_str - Parameter für die verteilte Verarbeitung 699
12 Administrationskommandos - KDCADM 701
12.1 KDCAPPL - Eigenschaften und Grenzwerte für den Betrieb ändern 704
12.2 KDCBNDL - Master-LTERMs austauschen 716
12.3 KDCDIAG - Diagnosehilfen ein- und ausschalten 717
12.4 KDCHELP - Syntax der Administrationskommandos abfragen 726
12.5 KDCINF - Informieren über Objekte und Anwendungsparameter 727
12.5.1 KDCINF Syntax-Beschreibung 729
12.5.2 Ausgabe von KDCINF 738
12.6 KDCLOG - Benutzer-Protokolldatei umschalten 771
12.7 KDCLPAP - Verbindungen zu (OSI-)LPAP-Partnern administrieren 772
12.8 KDCLSES - Verbindungen für LU6.1-Sessions auf-/abbauen 779
12.9 KDCLTAC - Eigenschaften von LTACs ändern 781
12.10 KDCLTERM - Eigenschaften von LTERM-Partnern ändern 783
12.11 KDCMUX - Eigenschaften von Multiplex-Anschlüssen ändern (BS2000-Systeme) 786
12.12 KDCPOOL - LTERM-Pools administrieren 789
12.13 KDCPROG - Lademodule/Shared Objects/DLLs austauschen 791
12.14 KDCPTERM - Eigenschaften von Clients und Druckern ändern 798
12.15 KDCSEND - Nachricht an LTERM-Partner senden (BS2000- Systeme) . . 803
12.16 KDCSHUT - Anwendungslauf beenden 804
12.17 KDCSLOG - SYSLOG-Datei administrieren 807
12.18 KDCSWTCH - Zuordnung Clients, Drucker zu LTERM- Partnern ändern . 812
12.19 KDCTAC - Transaktionscodes und TAC-Queues sperren, wieder freigeben . 816
12.20 KDCTCL - Prozess-Anzahl einer TAC-Klasse ändern 818
12.21 KDCUSER - Benutzereigenschaften ändern 824
13 Message Queues administrieren, Druckausgabe steuern 826
13.1 Berechtigungskonzept (BS2000-, Unix- und Linux-Systeme) 828
13.2 Message Queues administrieren (DADM) 831
13.2.1 Informieren über Nachrichten in einer Queue - DADM RQ 833
13.2.2 Benutzerinformation zu einer Nachricht lesen - DADM UI 834
13.2.3 Nachrichten in der Queue vorziehen - DADM CS 835
13.2.4 Nachrichten aus einer Queue löschen - DADM DA/DL 836
13.2.5 Nachrichten der Dead Letter Queue verschieben - DADM MA/MV 837
13.3 Drucker administrieren und Druckausgaben steuern (PADM) 838

13.3.1 Drucker administrieren mit PADM 839
13.3.1.1 Informationen über Drucker abfragen - PADM PI 840
13.3.1.2 Druckerstatus ändern - PADM CS 841
13.3.1.3 Drucker einem anderen LTERM-Partner zuordnen - PADM CA 842
13.3.2 Drucksteuerung mit PADM 843
13.3.2.1 Quittungsmodus ein- oder ausschalten - PADM AC/AT 845
13.3.2.2 Druckausgabe bestätigen oder wiederholen - PADM OK/PR 846
13.3.2.3 Informationen über zu quittierende Druckaufträge abfragen - PADM AI . 847
13.3.3 Behandlung von Fehlern bei der Druckausgabe 848
13.4 UTM-Teilprogramme für DADM- und PADM-Funktionen 849
13.4.1 KDCDADM und KDCPADM generieren 850
13.4.2 KDCDADM - Nachrichten administrieren 851
13.4.2.1 DELETE - Nachricht aus Message Queue löschen 852
13.4.2.2 INFORM - Über Message Queues und Nachrichten informieren 854
13.4.2.3 MOVE - Nachrichten aus der Dead Letter Queue verschieben 857
13.4.2.4 NEXT - Nachrichten in der Message Queue vorziehen 859
13.4.3 KDCPADM - Drucksteuerung und Drucker-Administration 860
13.4.3.1 INFORM - Informieren über Drucker eines Druckersteuer-LTERMs 861
13.4.3.2 MODE - Quittungsmodus eines Druckers ändern 864
13.4.3.3 PRINT - Druckausgabe bestätigen / wiederholen 865
13.4.3.4 STATE - Status eines Druckers ändern 866
13.4.3.5 SWITCH - Zuordnung Drucker zu LTERM-Partner ändern 867
14 Anhang 868
14.1 Programmschnittstelle zur Administration in COBOL 869
14.1.1 COPY-Elemente für die Programmschnittstelle in COBOL 870
14.1.2 KDCADMI-Funktionsaufruf 874
14.1.3 Hinweise zur Programmierung 875
14.2 Beispielprogramme 876
14.2.1 Das C-Teilprogramm HNDLUSR (BS2000-Systeme) 877
14.2.2 Das C-Teilprogramm SUSRMAX 878
14.2.3 Das COBOL-Teilprogramm COBUSER 879
14.2.4 Das C-Teilprogramm ENCRADM 880
14.2.5 Das C-Teilprogramm ADJTCLT 881
14.3 CALLUTM - Tool für Administration und Client-Server-Kommunikation (BS2000-Systeme) 886
14.3.1 Generierung 887
14.3.2 Beschreibung der CALLUTM-Programm-Anweisungen 890
14.3.3 Bestandteile, Systemumgebung, Softwarekonfiguration auf dem BS2000-System 906
14.3.4 Integration in eine UTM-Anwendung auf dem BS2000-System 907
14.3.5 Programmüberwachende Jobvariable auf dem BS2000-System 908

14.3.6 Meldungen von CALLUTM (BS2000-Systeme) 910
15 Fachwörter 913
16 Abkürzungen 954
17 Literatur 960

Anwendungen administrieren

1 Einleitung

Die IT-Infrastruktur heutiger Unternehmen als Herzstück und Motor des Geschäftes muss den Anforderungen des digitalen Zeitalters gerecht werden. Dabei muss sie mit vermehrten Datenmengen genauso zurechtkommen wie mit verschärften Anforderungen aus dem Umfeld, z.B. Einhaltung von Compliance-Vorgaben. Ebenso muss die Möglichkeit der kurzfristigen Integration weiterer Applikationen gegeben sein. Und alles dies unter dem Gesichtspunkt einer gewährleisteten Sicherheit.
Somit bestehen wesentliche Anforderungen an eine moderne IT-Infrastruktur u.a. aus

- Flexibilität und schier grenzenloser Skalierbarkeit auch für zukünftige Anforderungen
- hohe Robustheit bei höchster Verfügbarkeit
- absoluter Sicherheit in allen Belangen
- Anpassbarkeit an individuelle Bedürfnisse
- Verursachen geringer Kosten

Fujitsu bietet zur Bewältigung dieser Herausforderungen ein umfangreiches Portfolio innovativer Enterprise Hardware, Software und Support Services im Umfeld unserer Enterprise Mainframe Plattformen an und ist damit Ihr

- verlässlicher Service Provider, der Sie langfristig, flexibel und innovativ beim Betrieb der Mainframe-basierten Kernanwendungen Ihres Geschäftes unterstützt,
- optimaler Partner für die gemeinsame Abdeckung der Anforderungen einer Digitalen Transformation und
- langfristiger Partner aufgrund kontinuierlicher Anpassung moderner Schnittstellen, die eine moderne IT Landschaft mit all ihren Anforderungen mit sich bringt.

Mit openUTM stellt Ihnen Fujitsu eine vielfach erprobte und bewährte Lösung aus dem Middleware-Bereich zur Verfügung.

Die High-End-Plattform für Transaktionsverarbeitung openUTM bietet eine Ablaufumgebung, die all diesen Anforderungen moderner unternehmenskritischer Anwendungen gewachsen ist, denn openUTM verbindet alle Standards und Vorteile von transaktionsorientierten Middleware-Plattformen und Message Queuing Systemen:

- Konsistenz der Daten und der Verarbeitung
- Hohe Verfügbarkeit der Anwendungen
- Hohen Durchsatz auch bei großen Benutzerzahlen, d.h. höchste Skalierbarkeit
- Flexibilität bezüglich Änderungen und Anpassungen des IT-Systems

Eine UTM-Anwendung auf Unix-, Linux- und Windows-Systemen kann auf einem einzelnen Rechner als stand-alone UTM-Anwendung oder auf mehreren Rechnern gleichzeitig als UTM-Cluster-Anwendung betrieben werden.

openUTM ist Teil des umfassenden Angebots von **openSEAS**. Gemeinsam mit der Oracle Fusion Middleware bietet openSEAS die komplette Funktionalität für Anwendungsinnovation und moderne Anwendungsentwicklung. Im Rahmen des Produktangebots **openSEAS** nutzen innovative Produkte die ausgereifte Technologie von openUTM:

- BeanConnect ist ein Adapter gemäß der Java EE Connector Architecture (JCA) und bietet den standardisierten Anschluss von UTM-Anwendungen an Java EE Application Server. Dadurch können bewährte Legacy-Anwendungen in neue Geschäftsprozesse integriert werden.

- Bestehende UTM-Anwendungen können unverändert ins Web übernommen werden. Mit dem UTM-HTTP Interface und dem Produkt WebTransactions stehen in openSEAS zwei Alternativen zur Verfügung, welche es ermöglichen, bewährte Host-Anwendungen flexibel in neuen Geschäftsprozessen und modernen Einsatzszenarien zu nutzen.

Die Produkte BeanConnect und WebTransactions werden im Leistungsüberblick kurz dargestellt. Für diese Produkte gibt es eigene Handbücher.

Wenn im Folgenden von Linux-System bzw. Linux-Plattform die Rede ist, dann ist darunter eine Linux-Distribution wie z.B. SUSE oder Red Hat zu verstehen.

Wenn im Folgenden von Windows-System bzw. Windows-Plattform die Rede ist, dann sind damit alle Windows-Varianten gemeint, auf denen openUTM zum Ablauf kommt.

Wenn im Folgenden von Unix-System bzw. Unix-Plattform die Rede ist, dann ist darunter ein Unix-basiertes Betriebssystem wie z.B. Solaris oder HP-UX zu verstehen.

1.1 Zielgruppe und Konzept des Handbuchs

Das vorliegende Handbuch „Anwendungen administrieren“ richtet sich an Administratoren und Generierer einer UTM-Anwendung sowie an Programmierer von Administrationsprogrammen. Es beschreibt die Programmschnittstelle zur Administration, mit der Sie eigene Administrationsprogramme erstellen können, die Kommandoschnittstelle zur Administration und die Möglichkeiten zur Administration von Message Queues.

Für das Verständnis des Handbuchs sind Kenntnisse über die Programmiersprache C sowie über openUTM erforderlich. Insbesondere sollten Sie das Generierungstool KDCDEF und die Programmschnittstelle KDCS kennen. Ergänzende Informationen finden Sie in den openUTM-Handbüchern „Anwendungen generieren“ und „Anwendungen programmieren mit KDCS“.

In den Kapiteln 2, 3, 8, 9 und 10 dieses Handbuchs finden Sie allgemeine Informationen über die UTM-Administration. Sie richten sich sowohl an den Programmierer eigener Administrationsprogramme als auch an Benutzer der Administrationsprogramme. Sie informieren z.B. über die verschiedenen Schnittstellen, die openUTM Ihnen zur Administration Ihrer UTM-Anwendung zur Verfügung stellt, enthalten Beispiele dafür, wie Sie die Administrationsfunktionen von openUTM einsetzen können, um einen performanten und dauerhaften Betrieb Ihrer Anwendung sicherzustellen, und zeigen die Möglichkeiten der zentralen und automatischen Administration auf. Kapitel 8 geht auch näher auf die Administration von UTM-Cluster-Anwendungen auf Unix-, Linux- und Windows-Systemen ein.

Kapitel 4, 5, 7 und 11 enthalten spezielle Informationen für Programmierer eigener Administrationsprogramme. Sie beschreiben detailliert den Aufbau von Administrationsprogrammen sowie das dynamische Eintragen und Löschen von Clients, Druckern, Services und Benutzerkennungen. Kapitel 11 enthält alle Aufrufe der C-Programmschnittstelle zur Administration und die C-Datenstrukturen der Schnittstelle. Es wird detailliert beschrieben, welche Administrationsfunktionen Sie mit Hilfe der Schnittstelle durchführen können.

Kapitel 6 und 12 wenden sich speziell an den Benutzer der Administrationskommandos. In Kapitel 6 finden Sie Informationen zur synchronen und asynchronen Administration über Administrationskommandos. In Kapitel 12 finden Sie die Beschreibung der Administrationskommandos und der Administrationsfunktionen, die Sie mit den Kommandos ausführen können.

Kapitel 13 enthält Informationen zur Administration der lokalen Message Queues und der Administration von Druckern über ein Druckersteuer-LTERM.

1.2 Wegweiser durch die Dokumentation zu openUTM

In diesem Abschnitt erhalten Sie einen Überblick über die Handbücher zu openUTM und zum Produktumfeld von openUTM.

1.2.1 openUTM-Dokumentation

Die openUTM-Dokumentation besteht aus Handbüchern, den Online-Hilfen für den grafischen Administrationsarbeitsplatz openUTM WinAdmin und das grafische Administrationstool WebAdmin sowie Freigabemitteilungen.

Es gibt Handbücher und Freigabemitteilungen, die für alle Plattformen gültig sind, sowie Handbücher und Freigabemitteilungen, die jeweils für BS2000-Systeme bzw. für Unix-, Linux- und Windows-Systeme gelten.

Sämtliche Handbücher sind im Internet verfügbar unter der Adresse https://bs2manuals.ts.fujitsu.com. Für die Plattform BS2000 finden Sie die Handbücher auch auf der Softbook-DVD.

Die folgenden Abschnitte geben einen Aufgaben-bezogenen Überblick über die Dokumentation zu openUTM V7.0.

Eine vollständige Liste der Dokumentation zu openUTM finden Sie im Literaturverzeichnis.

Einführung und Überblick

Das Handbuch **Konzepte und Funktionen** gibt einen zusammenhängenden Überblick über die wesentlichen Funktionen, Leistungen und Einsatzmöglichkeiten von openUTM. Es enthält alle Informationen, die Sie zum Planen des UTM-Einsatzes und zum Design einer UTM-Anwendung benötigen. Sie erfahren, was openUTM ist, wie man mit openUTM arbeitet und wie openUTM in die BS2000-, Unix-, Linux- und Windows-Plattformen eingebettet ist.

Programmieren

- Zum Erstellen von Server-Anwendungen über die KDCS-Schnittstelle benötigen Sie das Handbuch **Anwendungen programmieren mit KDCS für COBOL, C und C++**, in dem die KDCS-Schnittstelle in der für COBOL, C und C++ gültigen Form und die Programmschnittstelle UTM-HTTP beschrieben sind. Die KDCS-Schnittstelle umfasst sowohl die Basisfunktionen des universellen Transaktionsmonitors als auch die Aufrufe für verteilte Verarbeitung. Es wird auch die Zusammenarbeit mit Datenbanken beschrieben.
  Die Programm-Schnittstelle UTM-HTTP stellt Funktionen zur Verfügung, die für die Kommunikation mit HTTP-Clients verwendet werden können.
- Wollen Sie die X/Open-Schnittstellen nutzen, benötigen Sie das Handbuch **Anwendungen erstellen mit X/Open-Schnittstellen**. Es enthält die openUTM-spezifischen Ergänzungen zu den X/Open-Programmschnittstellen TX, CPI-C und XATMI sowie Hinweise zu Konfiguration und Betrieb von UTM-Anwendungen, die X/Open-Schnittstellen nutzen. Ergänzend dazu benötigen Sie die X/Open-CAE-Spezifikation für die jeweilige X/Open-Schnittstelle.
- Wenn Sie Daten auf Basis von XML austauschen wollen, benötigen Sie das Dokument **XML für openUTM**. Darin werden die C- und COBOL-Aufrufe beschrieben, die zum Bearbeiten von XML-Dokumenten benötigt werden.
- Für BS2000-Systeme gibt es Ergänzungsbände für die Programmiersprachen Assembler, Fortran, Pascal-XT und PL/1.

Konfigurieren

Zur Definition von Konfigurationen steht Ihnen das Handbuch **Anwendungen generieren** zur Verfügung. Darin ist beschrieben, wie Sie mit Hilfe des UTM-Tools KDCDEF sowohl für eine stand-alone UTM-Anwendung als auch für eine UTM-Cluster-Anwendung auf Unix-, Linux- und Windows-Systemen.

- die Konfiguration definieren,
- die KDCFILE erzeugen,
- und im Falle einer UTM-Cluster-Anwendung die UTM-Cluster-Dateien erzeugen.

Zusätzlich wird gezeigt, wie Sie wichtige Verwaltungs- und Benutzerdaten mit Hilfe des Tools KDCUPD in eine neue KDCFILE übertragen, z.B. beim Umstieg auf eine neue Version von openUTM oder nach Änderungen in der Konfiguration. Für eine UTM-Cluster-Anwendung wird außerdem gezeigt, wie Sie diese Daten mit Hilfe des Tools KDCUPD in die neuen UTM-Cluster-Dateien übertragen.

Binden, Starten und Einsetzen

Um UTM-Anwendungen einsetzen zu können, benötigen Sie für das betreffende Betriebssystem (BS2000- bzw. Unix-, Linux- oder Windows-Systeme) das Handbuch **Einsatz von UTM-Anwendungen**.
Dort ist beschrieben, wie man ein UTM-Anwendungsprogramm bindet und startet, wie man sich bei einer UTM-Anwendung an- und abmeldet und wie man Anwendungsprogramme strukturiert und im laufenden Betrieb austauscht. Außerdem enthält es die UTM-Kommandos, die dem Terminal-Benutzer zur Verfügung stehen. Zudem wird ausführlich auf die Punkte eingegangen, die beim Betrieb von UTM-Cluster-Anwendungen zu beachten sind.

Administrieren und Konfiguration dynamisch ändern

- Für das Administrieren von Anwendungen finden Sie die Beschreibung der Programmschnittstelle zur Administration und die UTM-Administrationskommandos im Handbuch **Anwendungen administrieren**. Es informiert über die Erstellung eigener Administrationsprogramme für den Betrieb einer stand-alone UTM-Anwendung oder einer UTM-Cluster-Anwendung sowie über die Möglichkeiten, mehrere UTM-Anwendungen zentral zu administrieren. Darüber hinaus beschreibt es, wie Sie Message Queues und Drucker mit Hilfe der KDCS-Aufrufe DADM und PADM administrieren können.
- Wenn Sie den grafischen Administrationsarbeitsplatz **openUTM WinAdmin** oder die funktional vergleichbare Web-Anwendung **openUTM WebAdmin** einsetzen, dann steht Ihnen folgende Dokumentation zur Verfügung:
  - Die **WinAdmin-Beschreibung** und die **WebAdmin-Beschreibung** bieten einen umfassenden Überblick über den Funktionsumfang und das Handling von WinAdmin/WebAdmin.
  - Das jeweilige **Online-Hilfesystem** beschreibt kontextsensitiv alle Dialogfelder und die zugehörigen Parameter, die die grafische Oberfläche bietet. Außerdem wird dargestellt, wie man WinAdmin bzw. WebAdmin konfiguriert, um stand-alone UTM-Anwendungen und UTM-Cluster-Anwendungen administrieren zu können.

> **i** Details zur Integration von openUTM WebAdmin in den SE Manager des SE Servers finden Sie im SE Server Handbuch **Bedienen und Verwalten**.

Testen und Fehler diagnostizieren

Für die o.g. Aufgaben benötigen Sie außerdem die Handbücher **Meldungen, Test und Diagnose** (jeweils ein Handbuch für Unix-, Linux- und Windows-Systeme und für BS2000-Systeme). Sie beschreiben das Testen einer UTM-Anwendung, den Inhalt und die Auswertung eines UTM-Dumps, das Meldungswesen von openUTM, sowie alle von openUTM ausgegebenen Meldungen und Returncodes.

openUTM-Clients erstellen

Wenn Sie Client-Anwendungen für die Kommunikation mit UTM-Anwendungen erstellen wollen, stehen Ihnen folgende Handbücher zur Verfügung:

- Das Handbuch **openUTM-Client für Trägersystem UPIC** beschreibt Erstellung und Einsatz von Client-Anwendungen, die auf UPIC basieren. Es zeigt auf, was beim Programmieren einer CPI-C-Anwendung zu beachten ist und welche Einschränkungen es gegenüber der Programmschnittstelle X/Open CPI-C gibt.

- Das Handbuch **openUTM-Client für Trägersystem OpenCPIC** beschreibt, wie man OpenCPIC installiert und konfiguriert. Es zeigt auf, was beim Programmieren einer CPI-C-Anwendung zu beachten ist und welche Einschränkungen es gegenüber der Programmschnittstelle X/Open CPI-C gibt.
- Für das mit **BeanConnect** ausgelieferte Produkt **openUTM-JConnect** existiert neben dem Handbuch eine Java-Dokumentation mit der Beschreibung der Java-Klassen.
- Das Handbuch **BizXML2Cobol** beschreibt, wie Sie bestehende Cobol-Programme einer UTM-Anwendung so erweitern können, dass sie als Standard-Web-Service auf XML-Basis genutzt werden können. Die Arbeit mit der grafischen Bedienoberfläche ist in der zugehörigen **Online-Hilfe** beschrieben.
- Sie können auch mit dem Software-Produkt WS4UTM (WebServices for openUTM) Services von UTM-Anwendungen als Web Services verfügbar machen. Dazu benötigen Sie das Handbuch **Web-Services für openUTM.** Die Arbeit mit der grafischen Bedienoberfläche ist in der zugehörigen **Online-Hilfe** beschrieben.

Kopplung mit der IBM-Welt

Wenn Sie aus Ihrer UTM-Anwendung mit Transaktionssystemen von IBM kommunizieren wollen, benötigen Sie außerdem das Handbuch **Verteilte Transaktionsverarbeitung zwischen openUTM und CICS-, IMS- und LU6.2-Anwendungen**. Es beschreibt die CICS-Kommandos, IMS-Makros und UTM-Aufrufe, die für die Kopplung von UTM-Anwendungen mit CICS- und IMS-Anwendungen benötigt werden. Die Kopplungsmöglichkeiten werden anhand ausführlicher Konfigurations- und Generierungsbeispiele erläutert. Außerdem beschreibt es die Kommunikation über openUTM-LU62, sowie dessen Installation, Generierung und Administration.

Dokumentation zu PCMX

Mit openUTM auf Unix-, Linux- und Windows-Systemen wird die Kommunikationskomponente PCMX ausgeliefert. Die Funktionen von PCMX sind in folgenden Dokumenten beschrieben:

- Handbuch CMX (Unix-Systeme) "Betrieb und Administration" für Unix- und Linux- Systeme
- Online-Hilfe zu PCMX für Windows-Systeme

1.2.2 Dokumentation zum openSEAS-Produktumfeld

Die Verbindung von openUTM zum openSEAS-Produktumfeld wird im openUTM-Handbuch **Konzepte und Funktionen** kurz dargestellt. Die folgenden Abschnitte zeigen, welche der openSEAS-Dokumentationen für openUTM von Bedeutung sind.

Integration von Java EE Application Servern und UTM-Anwendungen

Der Adapter BeanConnect gehört zur Produkt-Suite openSEAS. Der BeanConnect-Adapter realisiert die Verknüpfung zwischen klassischen Transaktionsmonitoren und Java EE Application Servern und ermöglicht damit die effiziente Integration von Legacy-Anwendungen in Java-Anwendungen.

Das Handbuch **BeanConnect** beschreibt das Produkt BeanConnect, das einen JCA 1.5- und JCA 1.6-konformen Adapter bietet, der UTM-Anwendungen mit Anwendungen auf Basis von Java EE, z.B. mit dem Application Server von Oracle, verbindet.

Web-Anbindung und Anwendungsintegration

Anstatt der UTM-HTTP-Programmschnittstelle können Sie alternativ auch das Produkt WebTransactions verwenden. Dann benötigen Sie die Handbücher zu WebTransactions. Die Dokumentation wird durch JavaDocs ergänzt.

1.2.3 Readme-Dateien

Funktionelle Änderungen und Nachträge der aktuellen Produktversion zu diesem Handbuch entnehmen Sie bitte ggf. den Produkt-spezifischen Readme-Dateien.

Readme-Dateien stehen Ihnen online bei dem jeweiligen Produkt zusätzlich zu den Produkthandbüchern unter https://bs2manuals.ts.fujitsu.com zur Verfügung. Für die Plattform BS2000 finden Sie Readme-Dateien auch auf der Softbook-DVD.

Informationen auf BS2000-Systemen

Wenn für eine Produktversion eine Readme-Datei existiert, finden Sie auf BS2000-Systemen die folgende Datei:

`SYSRME.<product>.<version>.<lang>`

Diese Datei enthält eine kurze Information zur Readme-Datei in deutscher oder englischer Sprache (<lang>=D/E). Die Information können Sie am Bildschirm mit dem Kommando `/SHOW-FILE` oder mit einem Editor ansehen. Das Kommando `/SHOW-INSTALLATION-PATH INSTALLATION-UNIT=<product>` zeigt, unter welcher Benutzerkennung die Dateien des Produkts abgelegt sind.

Ergänzende Produkt-Informationen

Aktuelle Informationen, Versions-, Hardware-Abhängigkeiten und Hinweise für Installation und Einsatz einer Produktversion enthält die zugehörige Freigabemitteilung. Solche Freigabemitteilungen finden Sie online unter https://bs2manuals.ts.fujitsu.com.

1.3 Änderungen in openUTM V7.0

Die folgenden Abschnitte gehen näher auf die Änderungen in den einzelnen Funktionsbereichen ein.

1.3.1 Neue Server-Funktionen

UTM als HTTP-Server

Eine UTM-Anwendung kann auch als HTTP-Server fungieren.

Als Methoden werden GET, PUT, POST und DELETE unterstützt. Neben HTTP wird auch der Zugang über HTTPS unterstützt.

Dazu wurden folgende Schnittstellen geändert:

- Generierung

  Alle Systeme:

  - KDCDEF-Anweisung BCAMAPPL:
    - Beim Operand T-PROT= mit Wert SOCKET gibt es eine zusätzliche Angabe zum Transportprotokoll:

      *USP: Auf Verbindungen dieses Zugriffspunktes soll das UTM-Socket-Protokoll verwendet werden.

      *HTTP: Auf Verbindungen dieses Zugriffspunktes soll das HTTP-Protokoll verwendet werden.

      *ANY: Auf Verbindungen dieses Zugriffspunktes werden sowohl das UTM-Socket-Protokoll als auch das HTTP-Protokoll unterstützt.

    - Beim Operand T-PROT= mit Wert SOCKET gibt es zusätzlich die Angabe zur Verschlüsselung:

      SECURE: Auf Verbindungen dieses Zugriffspunktes erfolgt die Kommunikation unter Verwendung von Transport Layer Security (TLS).

    - Neuer Operand USER-AUTH = *NONE | *BASIC. Hiermit kann angegeben werden, welchen Authentisierungsmechanismus HTTP-Clients für diesen Zugangspunkt verwenden müssen.
  - KDCDEF-Anweisung HTTP-DESCRIPTOR:
    Mit dieser Anweisung wird eine Abbildung des in einem HTTP-Request empfangenen Path auf einen TAC definiert und es können zusätzliche Verarbeitungsparameter angegeben werden.

  BS2000-Systeme:

  - KDCDEF-Anweisung CHAR-SET:
    Mit dieser Anweisung können jeder der von openUTM zur Verfügung gestellten vier Code-Konvertierungen von UTM jeweils bis zu vier Character-Set Namen zugeordnet werden.
- Programmierung
  - KDCS- Kommunikationsbereich (KB):
    Im Kopf des KDCS-Kommunikationsbereichs gibt es im Feld *kccp/KCCP* neue Werte für die Client-Protokolle HTTP, USP-SECURE und HTTPS.
  - KDCS-Aufruf INIT PU:
    - Die Version der Schnittstelle wurde auf 7 erhöht.
    - Um die verfügbare Information vollständig zu erhalten, muss im Feld KCLI der Wert 372 angegeben werden.
    - Neue Felder zur Anforderung (KCHTTP/http_info) und Rückgabe (KCHTTPINF/httpInfo) von HTTP-spezifischen Informationen.
- Administrationsschnittstelle KDCADMI
  - Die Datenstrukturversion von KDCADMI wurde auf Version 11 geändert (Feld *version_data* im Parameterbereich).

- Neue Struktur *kc_http_descriptor_str* im Identifikationsbereich für die Unterstützung des HTTP Deskriptors.
- Neue Struktur *kc_character_set_str* im Identifikationsbereich für die Unterstützung des HTTP Charactersets.
- Neue Felder *secure_soc* und *user_auth* in Struktur *kc_bcamappl_str* für die Unterstützung von HTTP Zugangspunkten.

- Programmschnittstelle UTM-HTTP
  Zusätzlich zum KDCS-Interface bietet UTM ein Interface zum Lesen und Schreiben von HTTP Protokollinformationen und zur Behandlung des HTTP Message Bodys.
  Im Folgenden werden die Funktionen des Interface kurz aufgelistet:
  - Funktion *kcHttpGetHeaderByIndex()*
    Diese Funktion liefert den Namen und Wert des HTTP-Header-Feldes für den angegebenen Index zurück.
  - Funktion *kcHttpGetHeaderByName()*
    Die Funktion liefert den Wert des über den Namen spezifizierten HTTP-Header-Feldes zurück.
  - Funktion *kcHttpGetHeaderCount()*
    Diese Funktion liefert die Anzahl der in dem HTTP-Request enthaltenen Header-Felder zurück, die vom Teilprogramm gelesen werden können .
  - Funktion *kcHttpGetMethod()*
    Diese Funktion liefert die HTTP-Methode des HTTP-Requests zurück.
  - Funktion *kcHttpGetMputMsg()*
    Diese Funktion liefert die vom Teilprogramm erzeugte MPUT-Nachricht zurück.
  - Funktion *kcHttpGetPath()*
    Diese Funktion liefert den mit KC_HTTP_NORM_UNRESERVED normierten HTTP- Path des HTTP-Requests zurück.
  - Funktion *kcHttpGetQuery()*
    Diese Funktion liefert die mit KC_HTTP_NORM_UNRESERVED normierte HTTP- Query des HTTP -Requests zurück.
  - Funktion *kcHttpGetRc2String()*
    Hilfsfunktion um ein Funktionsergebnis vom Typ enum in einen abdruckbaren null-terminierten String umzuwandeln.
  - Funktion *kcHttpGetReqMsgBody()*
    Diese Funktion liefert den Message Body des HTTP Requests zurück.
  - Funktion *kcHttpGetScheme()*
    Diese Funktion liefert das Schema des HTTP- Requests zurück.
  - Funktion *kcHttpGetVersion()*
    Diese Funktion liefert die Version des HTTP- Requests zurück .
  - Funktion *kcHttpPercentDecode()*
    Funktion zur Umwandlung von Zeichen in Prozent-Darstellung in Zeichenfolgen in normale Ein-Zeichen-Darstellung.
  - Funktion *kcHttpPutHeader()*
    Diese Funktion übergibt einen HTTP-Header für die HTTP-Response .
  - Funktion *kcHttpPutMgetMsg()*
    Diese Funktion übergibt eine Nachricht für das Teilprogramm, die mit MGET gelesen werden kann.
  - Funktion *kcHttpPutRspMsgBody()*
    Diese Funktion übergibt eine Nachricht für den Message Body der HTTP-Response.

  - Function *kcHttpPutStatus()*
    Diese Funktion übergibt einen HTTP-Statuscode für die HTTP-Response.

- Kommunikation über den Secure Socket Layer (SSL)
  BS2000-Systeme:

  - Ist für eine UTM-Anwendung ein BCAMAPPL mit T-PROT=(SOCKET,...,SECURE) generiert, dann wird beim Anwendungsstart von UTM eine zusätzliche Task mit einem Reverse Proxy gestartet, der für die Anwendung als TLS Termination Proxy fungiert und über den sämtliche SSL-Kommunikation abgewickelt wird.

  Unix-, Linux- und Windows-Systeme :

  - Für einen sicheren Zugang über TLS steht ein weiterer Netzprozess vom Typ *utmnetssl* zur Verfügung.
    Sind für eine UTM-Anwendung BCAMAPPL mit T-PROT=(SOCKET,...,SECURE) generiert, dann wird beim Anwendungsstart von UTM eine Anzahl von *utmnetssl* Prozessen gestartet. Die Anzahl dieser Prozesse ist abhängig vom Wert LISTENER-ID dieser BCAMAPPL Objekte. In einem *utmnetssl* Prozess wird für die zugeordneten BCAMAPPL Portnummern die gesamte TLS-Kommunikation abgewickelt.

Verschlüsselung

Die Verschlüsselungsfunktionalität in UTM zwischen einer UTM-Anwendung und einem UPIC-Client wurde überarbeitet. Dabei wurden Sicherheitslücken geschlossen, moderne Methoden aufgenommen und die Auslieferung wie folgt vereinfacht:

- UTM-CRYPT Variante
  Bisher stand die Verschlüsselungsfunktionalität in UTM nur zur Verfügung, wenn man das Produkt UTM-CRYPT installiert hatte. Mit UTM V7.0 ist dies nicht mehr erforderlich. Ab dieser Version wird über die Generierung bzw. zum Anwendungsstart entschieden, ob die Verschlüsselungsfunktionalität zum Einsatz kommt oder nicht.
- Security
  Bei der Kommunikation zwischen einer UTM-Anwendung und einem UPIC-Client wurde eine Sicherheitslücke behoben.

  ! Das hat zur Folge, dass verschlüsselte Kommunikation einer UTM-Anwendung V7.0 nur zusammen mit UPIC-Client Anwendungen ab UPIC V7.0 möglich ist!

- Verschlüsselung Level 5 (*Unix-, Linux- und Windows-Systeme):*
  KDCDEF-Anweisungen PTERM, TAC und TPOOL
  Beim Operanden ENCRYPTION-LEVEL gibt es einen zusätzlichen Level 5. Dabei wird zur Vereinbarung des Session-Keys das auf Elliptic Curves basierende Diffie-Hellman Verfahren verwendet und Ein-/Ausgabe-Nachrichten werden mit dem AES-GCM Algorithmus verschlüsselt.

OSI-TP Kommunikation und Portnummern

BS2000-Systeme:

- KDCDEF-Anweisung OSI-CON
  Der Operand LISTENER-PORT kann auch auf BS2000-Systemen angegeben werden.
- Administrationsschnittstelle KDCADMI
  In der Struktur *kc_osi_con_str* wird auch auf BS2000-Systemen im Feld *listener-port* die Portnummer angezeigt.

Subnetze

In einer UTM-Anwendung können auch auf BS2000-Systemen Subnetze generiert werden, um den Zugang zu UTM-Anwendungen auf definierte IP-Adressbereiche beschränken zu können. Zusätzlich kann die Namensauflösung per DNS gesteuert werden.

Dazu wurden folgende Schnittstellen geändert:

- Generierung
  BS2000-Systeme:
  - KDCDEF-Anweisung SUBNET:
    Die SUBNET-Anweisung kann auch auf BS2000-Systemen angegeben werden.

  Alle Systeme:
  - KDCDEF-Anweisung SUBNET:
    Mit RESOLVE-NAMES=YES/NO kann angegeben werden, ob nach einem Verbindungsaufbau eine Namensauflösung per DNS stattfinden soll oder nicht.

    Falls eine Namensauflösung erfolgt, dann wird über die Administrationsschnittstelle und in Meldungen der echte Prozessorname des Kommunikationspartners angezeigt. Andernfalls wird als Prozessorname die IP-Adresse der Komunikationspartners sowie der Name des in der Generierung definierten Subnetzes angezeigt.
- Administrationsschnittstelle KDCADMI
  Die Strukturen *kc_subnet_str* und *kc_tpool_str* enthalten ein neues Feld *resolve_names*.

Zugangsdaten für den XA-Datenbank-Anschluss

Ein modifizierter aber noch nicht aktivierter Benutzername für den XA-Datenbank-Anschluss kann per Administration (KDCADMI) gelesen werden:

- Operationscode KC_GET_OBJECT:
  Datenstruktur *kc_db_info_str*: Neues Feld *db_new_userid*.

Reconnect für den XA-Datenbank-Anschluss

Wird bei einer XA Aktion zur Steuerung der Transaktion entdeckt, dass die Verbindung zur Datenbank nicht mehr besteht, wird versucht die Verbindung zu erneuern und die XA Aktion zu wiederholen.

Nur falls dies nicht erfolgreich ist, werden der betroffene UTM Prozess und die UTM-Anwendung abnormal beendet . Bisher wurde bei jedem Verbindungsverlust zur XA Datenbank unmittelbar ohne erneuten Verbindungs-Versuch die UTM-Anwendung abnormal beendet.

Sonstige Änderungen

- XA-Meldungen
  Die Meldungen bzgl. der XA-Schnittstelle wurden jeweils um die Inserts UTM-Userid und TAC erweitert. Betroffen sind die Meldungen K204-K207, K212-K215 und K217-K218.
- UTM-Tool KDCEVAL
  Im TRACE 2 Satz von KDCEVAL wurde im WAITEND Record der Typ des letzten Auftrags (Börsen-Announcements) aufgenommen (ersten beiden Bytes abdruckbar).

1.3.2 Entfallene Server-Funktionen

Im Einzelnen wurden folgende Funktionen gestrichen:

- Dienstprogramm KDCDEF
  Mehrere Funktionen wurden gestrichen und können nicht mehr in KDCDEF generiert werden. Wenn sie dennoch angegeben werden, wird dies im KDCDEF-Lauf mit einem Syntaxfehler abgelehnt.
  - KDCDEF-Anweisung PTERM
    Operanden-Werte 1 und 2 für ENCRYPTION-LEVEL
  - KDCDEF-Anweisung TPOOL
    Operanden-Werte 1 und 2 für ENCRYPTION-LEVEL
  - KDCDEF-Anweisung TAC
    Operanden-Wert 1 für ENCRYPTION-LEVEL
- *BS2000-Systeme*
  - UTM-Cluster:
    Auf BS2000-Systemen werden UTM-Cluster-Anwendungen nicht mehr unterstützt.
- *Unix-, Linux- und Windows-Systeme*
  - TNS Betrieb:
    Beim Start einer UTM-Anwendung wird die TNS-Generierung nicht mehr gelesen. Die Adressierungsinformation muss vollständig bei der Konfiguration mit KDCDEF hinterlegt werden.

1.3.3 Neue Client-Funktionen

Verschlüsselung

Die Verschlüsselungsfunktionalität in openUTM-Client wurde überarbeitet. Dabei wurden Sicherheitslücken geschlossen, moderne Methoden aufgenommen und die Auslieferung wie folgt vereinfacht:

- UTM-CLIENT-CRYPT Variante
  Bisher stand die Verschlüsselungsfunktionalität in openUTM-Client nur zur Verfügung, wenn man das Produkt UTM-CLIENT-CRYPT installiert hatte. Mit openUTM-Client V7.0 ist dies nicht mehr erforderlich. Ab dieser Version wird zum Ablaufzeitpunkt entschieden ob die Verschlüsselungsfunktionalität zum Einsatz kommt oder nicht.
- Security
  Bei der Kommunikation mit einer UTM-Anwendung wurde eine Sicherheitslücke behoben.
- Verschlüsselung Level 5
  openUTM-Client V7.0 unterstützt die Kommunikation mit UTM V7.0 Anwendungen, bei denen für die Verbindungen zum UPIC-Client ENCRYPTION-LEVEL 5 generiert wurde.
  Bei Level 5 wird zur Vereinbarung des Session-Keys das auf Elliptic Curves basierende Diffie-Hellman Verfahren verwendet und Ein-/Ausgabe-Nachrichten werden mit dem AES-GCM Algorithmus verschlüsselt. AES-GCM unterstützt die Authentifikation und die Verschlüsselung von Nachrichten.

  Der Level 5 wird von openUTM-Client auf allen Plattformen unterstützt.
- Verschlüsselung BS2000
  openUTM-Client (BS2000) verwendet analog zu Unix-, Linux- und Windows-Systemen openSSL anstatt BS2000-CRYPT.

1.3.4 Neue Funktionen für openUTM WinAdmin

WinAdmin unterstützt alle Neuerungen der openUTM V7.0 bzgl. der Programmschnittstelle zur Administration.

1.3.5 Neue Funktionen für openUTM WebAdmin

WebAdmin unterstützt alle Neuerungen der openUTM V7.0 bzgl. der Programmschnittstelle zur Administration.

1.4 Darstellungsmittel

Metasyntax

Die in diesem Handbuch verwendete Metasyntax können Sie der folgenden Tabelle entnehmen:

Formale Darstellung	Erläuterung	Beispiel
GROSSBUCHSTABEN	Großbuchstaben bezeichnen Konstanten (Namen von Aufrufen, Anweisungen, Feldnamen, Kommandos und Operanden etc.), die in dieser Form anzugeben sind.	`LOAD-MODE=STARTUP`
kleinbuchstaben	In Kleinbuchstaben sind in Syntaxdiagrammen und Operandenbeschreibung die Platzhalter für Operandenwerte dargestellt.	`KDCFILE=filebase`
kleinbuchstaben	Im Fließtext werden Variablen sowie Namen von Datenstrukturen und Feldern in kursiven Kleinbuchstaben dargestellt.	*utm-installationsverzeichnis* ist das UTM-Installationsverzeichnis
`Schreibmaschinenschrift`	In Schreibmaschinenschrift werden im Fließtext Kommandos, Dateinamen, Meldungen und Beispiele ausgezeichnet, die in genau dieser Form eingegeben werden müssen bzw. die genau diesen Namen oder diese Form besitzen.	Der Aufruf `tpcall`
{ } und |	In geschweiften Klammern stehen alternative Angaben, von denen Sie eine auswählen müssen. Die zur Verfügung stehenden Alternativen werden jeweils durch einen Strich getrennt aufgelistet.	`STATUS={ ON | OFF }`
[ ]	In eckigen Klammern stehen wahlfreie Angaben, die entfallen können.	`KDCFILE=( filebase [, { SINGLE | DOUBLE } ] )`
( )	Kann für einen Operanden eine Liste von Parametern angegeben werden, sind diese in runde Klammern einzuschließen und durch Kommata zu trennen. Wird nur ein Parameter angegeben, kann auf die Klammern verzichtet werden.	`KEYS=(key1, key2,...key`*n*`)`
Unterstreichen	Unterstreichen kennzeichnet den Standardwert.	`CONNECT= { YES | NO }`
Kurzform	Die Standardkurzform für Anweisungen, Operanden und Operandenwerte wird „fett“ hervorgehoben. Die Kurzform kann alternativ angegeben werden.	`TRANSPORT -SELECTOR =c‘C‘`

Formale Darstellung	Erläuterung	Beispiel

`...`	Punkte zeigen die Wiederholbarkeit einer syntaktischen Einheit an.
	Außerdem kennzeichnen die Punkte Ausschnitte aus einem Programm, einer Syntaxbeschreibung o.ä.
		`KDCDEF starten`
		`...`
		`OPTION`
		`DATA=statement_file`
		`...`
		`END`

Symbole

für Verweise auf umfassende und detaillierte Informationen zum jeweiligen Thema.

 für Hinweistexte.

 für Warnhinweise.

Sonstiges

utmpfad bezeichnet auf Unix-, Linux- und Windows-Systemen das Verzeichnis, unter dem openUTM installiert wurde.

filebase bezeichnet auf Unix-, Linux- und Windows-Systemen das Dateiverzeichnis der UTM-Anwendung. Dies ist der Basisname, der in der KDCDEF-Anweisung MAX KDCFILE= generiert wurde.

$userid bezeichnet auf BS2000-Systemen die Kennung, unter der openUTM installiert wurde.

upic-dir bezeichnet auf Unix-, Linux- und Windows-Systemen das Verzeichnis, unter dem openUTM-Client für Trägersystem UPIC installiert ist.

2 Überblick über die openUTM-Administration

Unter dem Begriff „Administration“ sind alle Aktivitäten zusammengefasst, die zur Steuerung und Verwaltung der laufenden Anwendung dienen. „Administrieren“ heißt, die Anwendung an geänderte Verhältnisse und Anforderungen anzupassen, ohne den Anwendungslauf zu unterbrechen.

Zur Administration Ihrer UTM-Anwendung stellt Ihnen openUTM folgende Schnittstellen und Tools zur Verfügung:

- die Kommandoschnittstelle, an der die Basis-Administrationsfunktionen zur Verfügung stehen. Sie ist realisiert im Administrationsprogramm KDCADM.
- die Programmschnittstelle KDCADMI für die Administration, mit deren Hilfe Sie eigene, speziell auf Ihre Anwendung zugeschnittene Administrationsprogramme erstellen können. An der Programmschnittstelle stehen Ihnen alle UTM-Administrationsfunktionen zur Verfügung.
- die Aufrufe PADM und DADM der Programmschnittstelle KDCS, mit denen Sie lokale Message Queues und Drucker administrieren sowie die Ausgabe von Druckaufträgen steuern können. Die UTM-Teilprogramme KDCDADM und KDCPADM stellen Ihnen alle Funktionen der KDCS-Aufrufe DADM und PADM zur Verfügung.
- die openUTM-Komponente WinAdmin, mit der Sie vom PC aus über eine grafische Oberfläche UTM-Anwendungen im Netz administrieren können.
- die openUTM-Komponente WebAdmin, die eine Web-Anwendung zur Administration von UTM-Anwendungen zur Verfügung stellt.

Nur auf BS2000-Systemen

- WebAdmin kann als Add-on in den SE Manager integriert werden.
- das Tool CALLUTM, mit dem Sie aus einer BS2000-Task heraus in UTM-Anwendungen auch Administrations-Vorgänge starten und Administrationskommandos aufrufen können.
- die Kommandos KDCISAT und KDCMSAT (Dialog-Transaktionscodes), mit denen Sie die SAT-Protokollierung für Ihre Anwendung steuern können. Diese Kommandos sind im openUTM-Handbuch „Einsatz von UTM-Anwendungen auf BS2000-Systemen“ beschrieben.

An der Kommandoschnittstelle und den Programmschnittstellen KDCADMI und KDCS bietet Ihnen openUTM umfassende Administrationsfunktionen, die einen performanten und flexiblen Einsatz der Anwendung ermöglichen und den unterbrechungsfreien Betrieb der Anwendung (7*24-Stunden-Betrieb) sicherstellen. Sie können z.B. folgende Aktionen durchführen:

- Die Performance der Anwendung kontrollieren, indem Sie Informationen über die aktuelle Auslastung der Anwendung abfragen, Leistungsengpässe und Fehler diagnostizieren und ggf. Maßnahmen ergreifen, die die Performance verbessern.
- Teile des Anwendungsprogramms bzw. das gesamte Anwendungsprogramm im laufenden Betrieb austauschen. So können Sie während des Anwendungslaufs Teilprogramme der Anwendung modifizieren bzw. neue Teilprogramme hinzufügen.
- Bei Hardwarefehlern an Terminals und Druckern deren Wiederanlaufinformationen bzw. Drucker-Queues einem anderen Terminal bzw. Drucker zuordnen. Damit kann der Benutzer von einem anderen Terminal aus die Arbeit fortsetzen bzw. die Druckausgaben können auf einen intakten Drucker umgeleitet werden.
- Clients, Drucker, LTERM-Pools, Benutzerkennungen, Services und die Anschlusspunkte der Kommunikationspartner (LTERM-, LPAP- und OSI-LPAP-Partner) bei Bedarf sperren und wieder freigeben.
- Die Verbindungen zu Clients, Druckern und Partner-Anwendungen auf- und abbauen bzw. auf Ersatzverbindungen umschalten.

- Informationen über die Konfiguration der Anwendung und die aktuelle Einstellung der Anwendungs- und Betriebsparameter anfordern.
- Die Konfiguration der Anwendung im laufenden Betrieb ändern, indem Sie Services, Benutzerkennungen, Clients, Drucker, Verbindungen und Sessionnamen für die verteilte Verarbeitung über LU6.1, Keysets und Transaktionscodes für Partner-Anwendungen neu in die Konfiguration aufnehmen bzw. aus der Konfiguration löschen.
- TAC-, USER- und temporäre Queues sowie die lokalen Message Queues von LTERM-Partnern und Transaktionscodes verwalten.
- Die Anwendung beenden.

Die Administrationsfunktionen von openUTM (mit Ausnahme der SAT-Administrationskommandos) können Sie sowohl im Dialog als auch über Message Queuing aufrufen. Die Administration über Message Queuing ermöglicht die automatische Administration der Anwendung über „programmierte Administratoren“, d.h. Sie können Programme erstellen, die zu einem bestimmten Zeitpunkt (Aufruf über DPUT) oder beim Eintreffen bestimmter Ereignisse Administrationsfunktionen ausführen. Die Aufrufe der Programmschnittstellen und die Administrationskommandos können insbesondere vom Event-Service MSGTAC aufgerufen werden.

Sie können auch die Möglichkeiten nutzen, die Ihnen die Benutzer-spezifischen Meldungsziele bieten. Mit Hilfe dieser Meldungsziele lassen sich Meldungen wie Nachrichten in einer TAC- oder USER-Queue über die KDCS-Programmschnittstelle mit der Funktion DGET lesen. Mit dieser Funktion und entsprechender Folgeverarbeitung können Sie MSGTAC-ähnliche Programme entwerfen, die spezifisch auf eine Nachricht reagieren.

Informationen zur automatischen Administration finden Sie im Kapitel „Administration automatisieren".

Für alle Administrationsfunktionen, die schreibend auf die Konfigurationsdaten der Anwendung zugreifen, ist die UTM-Administrationsberechtigung erforderlich. Daneben existiert eine schwächere Berechtigungsstufe, die zur Verwendung von Administrationsfunktionen berechtigt, die nur lesend auf die Anwendungsdaten zugreifen.

Zum Berechtigungskonzept siehe im Kapitel „Zugriffsrechte und Zugriffsschutz".

Im Folgenden finden Sie einen Überblick über den Funktionsumfang der einzelnen Schnittstellen und Tools, die Unterschiede zwischen ihnen und ihre jeweiligen Anwendungsgebiete.

2.1 Kommandoschnittstelle

Mit openUTM wird das Standard-Administrationsprogramm KDCADM ausgeliefert, in dem die Kommandoschnittstelle zur Administration realisiert ist. Die Kommandoschnittstelle zur Administration unterstützt einen Teil der Funktionen der Programmschnittstelle zur Administration (KDCADMI).

KDCADM stellt die Basis-Administrationsfunktionen zur Verfügung, die Sie benötigen, um die dauernde Verfügbarkeit der Anwendung zu gewährleisten und die Performance der Anwendung zu kontrollieren. Das dynamische Eintragen neuer Objekte und das Löschen von Objekten aus der Konfiguration ist mit KDCADM nicht möglich.

Um die einzelnen Funktionen von KDCADM aufrufen zu können, müssen Sie dem Programm KDCADM vorgegebene Transaktionscodes zuordnen. Diese Transaktionscodes werden Administrationskommandos genannt.

Für jede Funktion von KDCADM gibt es jeweils einen Dialog-Transaktionscode (Dialog-Kommando) und einen Asynchron-Transaktionscode (Asynchron-Kommando). Die Administrationsfunktionen von KDCADM können Sie also synchron im Dialog aufrufen oder asynchron über Message Queuing nutzen.

Beim Aufruf eines Kommandos können Sie Operanden angeben. Über diese Operanden definieren Sie die Art der Aktion, die ausgeführt werden soll, und auf welche Objekte der Anwendung sich die Aktion beziehen soll. Die Operanden sind für die jeweiligen Dialog- und Asynchron-Kommandos identisch.

Die KDCADM-Administrationskommandos und ihre Operanden sind im Kapitel „Administrationskommandos - KDCADM" beschrieben.

Die Eingabe der Administrationskommandos ist nur im Linemode möglich. Die Ausgabe erfolgt ebenfalls im Linemode. Der Einsatz von Formaten ist nicht möglich.

Informationen zum Layout der Ausgabe bei der Administration über Message Queuing finden Sie im Kapitel „Administration über Kommandos".

Die Administrationskommandos, die Sie im Betrieb nutzen wollen, und das Administrationsprogramm KDCADM müssen Sie explizit in die Konfiguration Ihrer Anwendung eintragen, entweder bei der KDCDEF-Generierung oder dynamisch mit Hilfe der Programmschnittstelle KDCADMI. Das Kommando KDCSHUT, mit dem Sie die Anwendung definiert beenden können, müssen Sie immer in die Konfiguration Ihrer Anwendung aufnehmen.

In der folgenden Tabelle finden Sie einen Überblick über die Funktionen von KDCADM sowie die Kommandos, mit denen die Funktionen aufgerufen werden.

Administrationsfunktion von KDCADM	**Dialog-Kommando**	**Asynchron-Kommando**
Einstellung der Anwendungsparameter und Timer ändern, aktuelle Prozesszahlen für die Anwendung festlegen, Verbindungsaufbau zu den Druckern, für die Druckaufträge existieren, das gesamte Anwendungsprogramm austauschen	KDCAPPL	KDCAPPLA
Master-LTERMs zweier LTERM-Bündel austauschen	KDCBNDL	KDCBNDLA
Diagnoseunterlagen erstellen, z.B. UTM-Diagnose-Dump anfordern	KDCDIAG	KDCDIAGA
Eigenschaften der Objekte und aktuelle Einstellung der Anwendungsparameter abfragen, Statistikinformationen anfordern	KDCINF	KDCINFA

Administrationsfunktion von KDCADM	**Dialog-Kommando**	**Asynchron-Kommando**
Benutzer-Protokolldatei auf die nächste Dateigeneration umschalten	KDCLOG	KDCLOGA
LTERM-Partner (ent-)sperren, Verbindungen auf- und abbauen	KDCLTERM	KDCLTRMA
Anzahl der für einen LTERM-Pool zugelassenen Clients ändern	KDCPOOL	KDCPOOLA
Lademodule/Shared Objects/DLLs der Anwendung austauschen	KDCPROG	KDCPROGA
Clients/Drucker (ent-)sperren, Verbindungen auf- und abbauen	KDCPTERM	KDCPTRMA
UTM-Anwendungslauf beenden	KDCSHUT	KDCSHUTA
System-Protokolldatei (SYSLOG) der Anwendung umschalten, Größenüberwachung ein- und ausschalten, Schwellwert für die Größenüberwachung ändern, Informationen über SYSLOG abfragen	KDCSLOG	KDCSLOGA
Zuordnungen von Client/Drucker zu LTERM-Partner ändern	KDCSWTCH	KDCSWCHA
Transaktionscodes (lokale Services) (ent-)sperren	KDCTAC	KDCTACA
Anzahl der Prozesse ändern, die maximal gleichzeitig Aufträge für eine TAC-Klasse bearbeiten	KDCTCL	KDCTCLA
Benutzerkennungen (ent-)sperren, Passwörter ändern	KDCUSER	KDCUSERA
Nur auf BS2000-Systemen:		
Anwendungsteile im Common Memory Pool austauschen, die für den Austausch vorgemerkt sind.	KDCAPPL	KDCAPPLA
Multiplexanschlüsse (ent-)sperren, Verbindungen auf- und abbauen	KDCMUX	KDCMUXA
Nachricht an ein oder mehrere Dialog-Terminals senden	KDCSEND	KDCSENDA
Zur Administration der Server-Server-Kommunikation über LU6.1 und OSI TP gibt es folgende Funktionen:		
Logische Verbindungen zu Partner-Anwendungen auf- und abbauen, Ersatzverbindungen zu OSI TP-Partnern schalten, LPAP- bzw. OSI-LPAP-Partner (ent-)sperren, Timer zur Überwachung von Sessions und Associations ändern	KDCLPAP	KDCLPAPA
Logische Verbindungen für eine Session auf- und abbauen.	KDCLSES	KDCLSESA
Fernen Service (LTAC) für die lokale Anwendung (ent-)sperren, Timer zur Überwachung des Session/Association-Aufbaus und der Antwortzeiten einstellen.	KDCLTAC	KDCLTACA

Funktionen und Transaktionscodes von KDCADM

2.2 Programmschnittstelle KDCADMI

Mit der Programmschnittstelle zur Administration (KDCADMI) können Sie Administrationsprogramme erstellen, die speziell auf Ihre Anwendung zugeschnitten sind. Diese Programmschnittstelle wird in C/C++ und COBOL zur Verfügung gestellt. In diesem Handbuch wird die Programmschnittstelle für C/C++ beschrieben. Da die COBOL-Schnittstelle weitgehend der C/C++-Schnittstelle entspricht, können Sie die Ausführungen in diesem Handbuch auch bei der Erstellung von COBOL-Administrationsprogrammen zu Rate ziehen. Zusätzliche Informationen, die Sie für die Erstellung von Administrationsprogrammen in COBOL benötigen, finden Sie im Abschnitt "Programmschnittstelle zur Administration in COBOL".

Die Programmschnittstelle bietet Ihnen Funktionen, die über die Basis-Administrationsfunktionen von KDCADM hinausgehen. Folgende Funktionen stehen Ihnen an der Programmschnittstelle KDCADMI zusätzlich zur Verfügung:

- Funktionen, mit denen Sie die Konfiguration dynamisch ändern können:
  Sie können neue Services (Teilprogramme, Transaktionscodes), Clients, Drucker, Benutzerkennungen, Verbindungen und Sessionnamen für die verteilte Verarbeitung über LU6.1, Keysets, Transaktionscodes für Partner-Anwendungen und Service-gesteuerte Queues dynamisch in die Konfiguration aufnehmen, aus der Konfiguration löschen oder Eigenschaften von Objekten oder Anwendungsparametern verändern.
- Inverser KDCDEF:
  Sie können aus den Konfigurationsinformationen, die in der KDCFILE gespeichert sind, Steueranweisungen für das Generierungstool KDCDEF erzeugen.
  Damit können Änderungen der Konfiguration, die während des Anwendungslaufs durchgeführt wurden, in eine Neugenerierung der Anwendung übernommen werden.
- Ausgabe aller Konfigurationsdaten bei der Informationsabfrage:
  Bei der Informationsabfrage zu einzelnen Objekten bzw. Anwendungsparametern werden alle Konfigurationsdaten zurückgeliefert, die in der KDCFILE zu diesem Objekt bzw. Parameter gespeichert sind. In einem selbst erstellten Administrationsprogramm können Sie genau die Daten auswerten und weiterverarbeiten, die in dem speziellen Anwendungsfall von Interesse sind. Bei der Informationsabfrage können Sie die Ausgabe auf Objekte beschränken, die besondere Kriterien erfüllen, indem Sie diese Selektionskriterien beim Aufruf angeben.

In der folgenden Tabelle sind die Funktionen von KDCADMI und die Operationscodes, über die Funktionen im Programm aufgerufen werden, aufgelistet.

Die Programmschnittstelle KDCADMI und alle Datenstrukturen sind im Kapitel „Programmschnittstelle zur Administration - KDCADMI" beschrieben.
Informationen zur dynamischen Administration und zum inversen KDCDEF finden Sie im Kapitel „Konfiguration dynamisch ändern" und „Generierungsanweisungen aus der KDCFILE erzeugen".

KDCADMI-Funktion	KDCADMI-Operationscode
Das gesamte Anwendungsprogramm austauschen, ohne die Anwendung herunterzufahren. *BS2000-Systeme:* Anwendungsteile im Common Memory Pool austauschen, die für den Austausch vorgemerkt sind. *Unix-, Linux- und Windows-Systemen:* Dabei geben Sie an, ob die nächsthöhere Version oder die nächstniedrigere Version oder die aktuelle Version des Anwendungsprogramms geladen werden soll.	KC_CHANGE_APPLICATION
UTM-Diagnose-Dump erzeugen, ohne die Anwendung zu beenden	KC_CREATE_DUMP
Die Konfiguration der Anwendung dynamisch um neue Services (Teilprogramme, Transaktionscodes), Clients, Drucker, Benutzerkennungen, Verbindungen und Sessionnamen für die verteilte Verarbeitung über LU6.1, Keysets, Transaktionscodes für Partner-Anwendungen und Service-gesteuerte Queues erweitern.	KC_CREATE_OBJECT
Inversen KDCDEF-Lauf online starten.	KC_CREATE_STATEMENTS
Clients, Drucker, Benutzerkennungen, Services, Verbindungen und Sessionnamen für die verteilte Verarbeitung über LU6.1, Keysets, Transaktionscodes für Partner-Anwendungen und Service-gesteuerte Queues aus der Konfiguration der Anwendung löschen.	KC_DELETE_OBJECT
RSA-Schlüsselpaar erzeugen, aktivieren, löschen. Public-Schlüssel eines RSA-Schlüsselpaares auslesen.	KC_ENCRYPT
Name und Eigenschaften der Objekte, aktuelle Einstellung der Anwendungsparameter abfragen, Statistikinformationen anfordern.	KC_GET_OBJECT
Unix-, Linux- und Windows-Systeme: Für alle oder einen einzelnen Benutzer, die /der noch an einer ausgefallenen Knoten-Anwendung als angemeldet vermerkt sind/ist oder die/der einen an die ausgefallene Knoten-Anwendung gebundenen Vorgang haben/hat, ein erneutes Anmelden ermöglichen. Sperre der Cluster-User-Datei nach nicht ordnungsgemäß beendetem KDCDEF-Lauf wieder aufheben. (Nur bei UTM-Cluster-Anwendungen)	KC_LOCK_MGMT

KDCADMI-Funktion	KDCADMI-Operationscode
Eigenschaften von Objekten oder Anwendungsparameter ändern, z.B.: Einstellung der Anwendungsparameter und Timer ändern,aktuelle Prozesszahlen für die Anwendung festlegen, Traces ein-/ausschalten,Lademodule/Shared Objects/DLLs der Anwendung austauschen,Benutzerkennungen, Transaktionscodes, Clients/Drucker oder Anschlüsse für Partner-Anwendungen (ent-)sperren, Verbindungen zu Clients, Druckern und Partner-Anwendungen auf- und abbauen, OSI TP-Ersatzverbindungen aktivieren, Anzahl der für einen LTERM-Pool zugelassenen Clients ändern, Zuordnungen von Client/Drucker zu LTERM-Partner ändern, Zähler für Statistikdaten zurücksetzen, Keys in Keysets ändern, Zugriffsschutz für Transaktionscodes, Benutzer und TAC-Queues ändern.	KC_MODIFY_OBJECT
Unix-, Linux- und Windows-Systeme: Anwendungsdaten aus einer beendeten in eine laufende Knoten-Anwendung importieren (nur bei UTM-Cluster-Anwendungen).	KC_ONLINE_IMPORT
Transaktion zurücksetzen, die sich im Zustand PTC (prepare to commit) befindet.	KC_PTC_TA
Nur auf BS2000-Systemen: Nachricht an ein Dialog-Terminal oder an alle aktiven Dialog-Terminals senden.	KC_SEND_MESSAGE
UTM-Anwendungslauf beenden.	KC_SHUTDOWN
Verbindung zu Druckern aufbauen, für die Druckaufträge existieren.	KC_SPOOLOUT
System-Protokolldatei (SYSLOG) der Anwendung umschalten, Größenüberwachung ein-/ ausschalten, Schwellwert für die Größenüberwachung ändern, Informationen über SYSLOG anfordern.	KC_SYSLOG
IP-Adressen der generierten Kommunikationspartner ermitteln; auf BS2000-Systemen: nur für T-PROT=SOCKET	KC_UPDATE_IPADDR
Die Benutzer-Protokolldatei(en) auf die nächste Dateigeneration umschalten.	KC_USLOG

Administrationsfunktionen der Programmschnittstelle zur Administration

Neben dem größeren Funktionsumfang, den Sie in selbst erstellten Administrationsprogrammen nutzen können, haben Administrationsprogramme, die die Funktionen der Programmschnittstelle nutzen, folgende Vorteile:

- Bei der Administration über Message Queuing können Sie den Empfänger der Ergebnisse frei wählen. Sie können also abhängig vom Ergebnis eines KDCADMI-Aufrufs verschiedene Folge-Transaktionen aufrufen. Daraus ergeben sich Vorteile für die automatische und programmierte Administration.
- Die Ergebnisse eines Administrationsaufrufs können in demselben Teilprogramm, das diesen Administrationsaufruf enthält, ausgewertet und weiterverarbeitet werden. Ein Administrationsprogramm kann beliebig viele Administrationsaufrufe enthalten. Die Anzahl von Administrationsaufrufen, die der Transaktionssicherung unterliegen und in einer gemeinsamen Transaktion ausgeführt werden sollen, ist allerdings durch die generierte Größe des Wiederanlaufbereichs beschränkt (Generierungsanweisung MAX, Parameter RECBUF, siehe openUTM-Handbuch „Anwendungen generieren“).

- Nur auf BS2000-Systemen: Für die Ein- und Ausgabe der Administrationsprogramme können Sie Formate einsetzen.

Die Aufrufe der Administrationsfunktionen müssen zwischen den KDCS-Aufrufen INIT und PEND erfolgen. Die für den Datenaustausch zwischen openUTM und dem Programm benötigten Datenstrukturen sind vordefiniert. Für C /C++ stehen die Datenstrukturen in der Include-Datei *kcadminc.h* (Unix-, Linux- und Windows-Systeme) bzw. im Include-Element *kcadminc.h* der Bibliothek SYSLIB.UTM.070.C (BS2000-Systeme).

Informationen zum Programmaufbau finden Sie im Kapitel „Erstellen eigener Administrationsprogramme".

In openUTM auf BS2000-, Unix-, Linux- oder Windows-Systemen werden identische Datenstrukturen verwendet. Die Datenstrukturen enthalten einige Felder, die nur für ein Betriebssystem relevant sind. Diese Felder müssen in einem anderen Betriebssystem mit binär null versorgt werden. In welchem Betriebssystem die jeweilige Anwendung abläuft, kann das Programm selbst mit Hilfe eines KDCADMI-Aufrufs ermitteln.

Da die Aufrufe von KDCADMI und die verwendeten Datenstrukturen plattform-neutral sind, können Sie mit KDCADMI Administrationsprogramme erstellen, die

- es erlauben, mehrere UTM-Anwendungen von einer „zentralen“ Stelle aus zu administrieren. Dabei kann es sich um UTM-Anwendungen auf verschiedenen Plattformen handeln. Insbesondere können Sie von einer UTM-Anwendung auf Unix-, Linux- oder Windows-Systemen aus UTM-Anwendungen auf BS2000-Systemen administrieren und umgekehrt. Die Anwendungen können unter verschiedenen Versionen von openUTM ablaufen.
- portierbar sind. Dieselbe Source eines Administrationsprogramms können Sie auf jeder der drei Plattformen übersetzen und dort in eine UTM-Anwendung einbinden.

Informationen zur zentralen Administration von Anwendungen finden Sie im Kapitel „Zentrale Administration mehrerer Anwendungen".

Die Aufrufe von KDCADMI können bis auf eine Ausnahme (Abbruch des Anwendungslaufs: KC_SHUTDOWN mit Subcode KC_KILL) in Dialog- und Asynchron-Vorgängen abgesetzt werden.

Die Dialog-Vorgänge können von Benutzern am Terminal, über UPIC-Clients und OpenCPIC-Partner, von einer Partner-Anwendung oder von HTTP-Clients gestartet werden.
Die Asynchron-Vorgänge können von Benutzern am Terminal, von Partner-Anwendungen und OpenCPIC-Partnern oder aus einem Teilprogramm heraus gestartet werden.

> **i** Die Programmschnittstelle zur Administration unterliegt der Kompatibilitätsgarantie, d.h. sie wird über einige Versionen von openUTM sourcekompatibel angeboten. Deshalb müssen Administrationsprogramme bei Versionsübergängen nicht angepasst werden, wenn diese als KDCADMI-Datenstruktur-Version die Version setzen, für die sie entwickelt wurden. D.h. die Administrationsprogramme können unverändert neu übersetzt und in eine UTM-Anwendung der Folgeversion eingebunden werden.

2.3 Beispielprogramme

Mit dem Produkt openUTM werden Beispielprogramme als Sourcecode und Objektmodule ausgeliefert, die Sie als Programmiervorlage für eigene Administrationsprogramme verwenden und nach Bedarf modifizieren, übersetzen und in Ihre Anwendung einbinden können. Bei den Beispielprogrammen handelt es sich um die Programme HNDLUSR (nur BS2000-Systeme), ENCRADM, SUSRMAX und COBUSER. Eine einführende Beschreibung finden Sie im Abschnitt „Beispielprogramme".

2.4 PADM, DADM zur Administration von Message Queues und Druckern

An der KDCS-Programmschnittstelle stehen die Aufrufe PADM und DADM zur Verfügung, mit denen Sie die Message Queues und Drucker der Anwendung administrieren und die Ausgabe am Drucker steuern können.

Sie können z.B. die Reihenfolge der Aufträge bzw. Nachrichten in einer Queue ändern, Aufträge bzw. Nachrichten aus den Queues löschen, Druckerbündel erzeugen und die Druckausgabe bei Defekt eines Druckers auf einen anderen Drucker umleiten. Außerdem können Sie Nachrichten der Dead Letter Queue in andere Message Queues verschieben, um sie zu verarbeiten.

Mit den Aufrufen PADM und DADM kann ein Benutzer oder Client ohne Administrationsberechtigung Drucker und deren Message Queues administrieren und die Ausgabe am Drucker steuern. Somit können „normale“ Benutzer ihre eigenen „lokalen“ Drucker und die an diese gerichteten Druckaufträge selbst administrieren. Die Administration muss dann von dem Druckersteuer-LTERM aus erfolgen, dem der zu administrierende Drucker zugeordnet ist.

PADM und DADM können auch von dem Event Service MSGTAC verwendet werden. Die MSGTAC-Routine kann z.B. bei Ausfall eines Druckers automatisch gestartet werden und mit PADM- und DADM-Aufrufen entsprechende Maßnahmen einleiten.

Mit openUTM werden die Teilprogramme KDCDADM und KDCPADM ausgeliefert. Diese Beispielprogramme machen alle Leistungen der Aufrufe DADM und PADM zugänglich, ohne dass Sie eigene Teilprogramme erstellen müssen.

Die Aufrufe PADM und DADM sowie die Programme KDCDADM und KDCPADM sind im Kapitel „Message Queues administrieren, Druckausgabe steuern" beschrieben.

Druckausgaben aus einer UTM-Anwendung heraus werden von openUTM auf Windows-Systemen nicht unterstützt. Aus diesem Grund ist die Funktion PADM in UTM-Anwendungen auf Windows-Systemen nicht relevant.

2.5 Das Tool CALLUTM (BS2000-Systeme)

CALLUTM ist ein UPIC-Client unter einem BS2000-System, mit dem Sie aus einer beliebigen BS2000-Task heraus UTM-Services aufrufen können. CALLUTM bietet eine SDF-Oberfläche und kann dazu eingesetzt werden, Administrations-Vorgänge in UTM-Anwendungen am gleichen Rechner oder auf anderen Rechnern im Netz zu starten. Insbesondere können Sie über CALLUTM mehrere UTM-Anwendungen im Netz zentral administrieren. Dabei kann es sich sowohl um UTM-Anwendungen auf BS2000-Systemen als auch auf Unix-, Linux- und Windows-Systemen handeln. CALLUTM ist sowohl im Dialog als auch im Batch-Betrieb ablauffähig.

CALLUTM ist im Anhang im Abschnitt "CALLUTM - Tool für Administration und Client-Server-Kommunikation (BS2000-Systeme)" beschrieben.

2.6 openUTM WinAdmin und openUTM WebAdmin

Die openUTM-Komponenten WinAdmin und WebAdmin stellen Ihnen komfortable grafische Oberflächen zur Administration einzelner oder auch mehrerer UTM-Anwendungen zur Verfügung.

WinAdmin und WebAdmin bieten im Wesentlichen den gleichen Funktionsumfang. Während openUTM WinAdmin eine Java-Anwendung ist, die auf Windows-, Unix- und Linux-Systemen läuft, ist openUTM WebAdmin eine Web-Anwendung, auf die von beliebigen Rechnern oder mobilen Geräten über einen Web-Browser zugegriffen werden kann.

Die UTM-Anwendungen können im Netz verteilt sein. Sie können auf allen freigegebenen Plattformen laufen und unterschiedliche Versionsstände besitzen. Sowohl WinAdmin als auch WebAdmin unterstützen den vollen Funktionsumfang der Programmschnittstelle, den die jeweilige openUTM-Version bietet.

Die zu administrierenden UTM-Anwendungen lassen sich zu Kollektionen gruppieren und können so gemeinsam administriert werden.

Damit eine UTM-Anwendung über WinAdmin oder WebAdmin administrierbar ist, müssen Sie in dieser Anwendung das Administrationsprogramm KDCWADMI und den zugehörigen Transaktionscode KDCWADMI generieren. Für den Transaktionscode müssen Sie ADMIN=YES angeben. Das Programm KDCWADMI gehört zum Lieferumfang von openUTM.

Mit WinAdmin und WebAdmin können Sie auch UTM-Anwendungen starten und beenden. Für das Starten von UTM-Anwendungen wird vorausgesetzt, dass auf den beteiligten Rechnern openFT im Einsatz ist. Daher kann das openUTM WebAdmin Add-on im SE Manager keine UTM-Anwendungen starten.

Security

Für die Administration über WinAdmin oder WebAdmin stehen Ihnen sämtliche UTM-Security-Funktionen zur Verfügung, angefangen vom Zugangsschutz durch UTM-Benutzerkennung und - Passwort bis zur Verschlüsselung von Passwort und Daten.

Darüber hinaus bieten WinAdmin und WebAdmin zusätzlich ein eigenes Benutzerkonzept. Sie können mehrere Benutzer definieren und mit unterschiedlichen Rechten ausstatten, angefangen vom Benutzer mit reinem Leserecht bis zum „Master“, dem Administrator von WinAdmin bzw. WebAdmin. Für jeden Benutzer wird der Zugang zu WinAdmin bzw. WebAdmin durch Passwort geschützt.

Unterschiede zwischen WinAdmin und WebAdmin

Mit WinAdmin ist es möglich, in einem Schritt Objekte mehrerer Anwendungen zu modifizieren oder mehrere Administrationsschritte in einer Transaktion zusammenzufassen.

Eine Einführung zu WinAdmin und WebAdmin finden Sie im Abschnitt „Administration über WinAdmin und WebAdmin".

3 Objekte administrieren, Parameter einstellen

Dieses Kapitel gibt einen Überblick über die Möglichkeiten, die Ihnen die UTM-Administration bietet. Es werden hier exemplarisch einige Anwendungsgebiete der UTM-Administration aufgeführt. Auf die Administrationskommandos und Aufrufe der Programmschnittstelle, mit denen Sie die einzelnen Aktionen ausführen können, wird nur verwiesen.

Eine umfassende Beschreibung der Aktionen, die Sie mit Hilfe der Programmschnittstelle und mit den Administrationskommandos insgesamt durchführen können, finden Sie im Kapitel „Programmschnittstelle zur Administration - KDCADMI" bzw. im Kapitel „Administrationskommandos - KDCADM".

Auf die Administrationsfunktionen zum dynamischen Eintragen neuer Objekte in die Konfiguration, zum Verändern von Objekteigenschaften und zum Löschen von Objekten wird in diesem Kapitel nicht eingegangen. Diese Administrationsfunktionen sind im Kapitel „Konfiguration dynamisch ändern" beschrieben.

In der folgenden Beschreibung werden folgende Symbole verwendet:

verweist auf das Administrationskommando, mit dem Sie die Aktionen ausführen können. Es wird immer nur das Dialog-Kommando angegeben. Sie können zur Ausführung der angegebenen Aktionen aber auch das entsprechende Asynchron-Kommando verwenden (siehe Tabelle im Abschnitt "Kommandoschnittstelle").

verweist auf den Funktionsaufruf an der Programmschnittstelle für die Administration, mit dem Sie die Administrationsfunktion ausführen können.

Alle in diesem Abschnitt angesprochenen Funktionen können Sie auch über die Administrations-Tools WinAdmin und WebAdmin nutzen.

3.1 Informationsfunktionen von openUTM

openUTM stellt Ihnen Informationsfunktionen zur Verfügung, mit denen Sie sich einen Überblick über die Konfiguration der Anwendung, die Einstellung der Anwendungsparameter und die momentane Auslastung der Anwendung verschaffen können. Die Informationsfunktionen der UTM-Administration können Sie aufrufen mit:

KDCINF

KC_GET_OBJECT

Die Informationsfunktionen können auch von Benutzern genutzt werden, die nicht administrationsberechtigt sind (siehe Kapitel „Zugriffsrechte und Zugriffsschutz").

Mit Hilfe der Informationsfunktionen können Sie sich z.B. folgende Informationen ausgeben lassen:

- Anwendungs- und Systemparameter, die bei der KDCDEF-Generierung mit der Anweisung MAX festgelegt wurden ("type=SYSPARM" im Abschnitt "Ausgabe von KDCINF" / "kc_max_par_str - Maximalwerte der Anwendung (MAX-Parameter)").
- Anzahl der Prozesse, die aktuell für die Anwendung aktiv sind, Anzahl der Prozesse, die maximal gleichzeitig für die Asynchron-Verarbeitung zur Verfügung stehen, Anzahl der Prozesse, die maximal gleichzeitig für die Bearbeitung von Services zur Verfügung stehen, die blockierende Aufrufe enthalten, z.B. den KDCS-Aufruf PGWT oder den XATMI-Aufruf tpcall ("type=SYSPARM" im Abschnitt "Ausgabe von KDCINF" / "kc_tasks_par_str - Anzahl der Prozesse").
- Daten zur aktuellen Auslastung der Anwendung. Das sind z.B. Auslastung des Pagepools bzw. Cluster-Pagepools, Anzahl der insgesamt ausgetauschten Nachrichten, Anzahl der angemeldeten Benutzer und Clients, Anzahl der derzeit offenen Vorgänge, Anzahl der pro Zeiteinheit durchgeführten Transaktionen, Anzahl der in den Message Queues gespeicherten Aufträge ("type=STATISTICS" und "type=SYSPARM" im Abschnitt "Ausgabe von KDCINF" / "kc_curr_par_str - Aktuelle Werte der Anwendungsparameter").
- Aktuelle Einstellung der Timer. In openUTM sind z.B. Timer definiert für das Belegen von und Warten auf Betriebsmittel, das Warten auf eine Antwort vom Dialog-Partner innerhalb und außerhalb einer Transaktion, das Warten auf Quittungen und das Warten auf den Aufbau einer Verbindung oder Session ("type=SYSPARM" im Abschnitt "Ausgabe von KDCINF" / "kc_timer_par_str - Timer-Einstellungen").
- Konfigurationsdaten zu allen Objekten, die in der Konfiguration enthalten sind. Das sind die Namen und logischen Eigenschaften, die beim Eintragen der Objekte in die Konfiguration festgelegt wurden. Dazu gehören u. a. auch die Schwellwerte für die Message Queues, die Anzahl der LTERM-Partner eines LTERM-Pools oder die generierte Maximalzahl paralleler Verbindungen zu einer OSI TP-Partner-Anwendung.
- Status der einzelnen Kommunikationspartner und Drucker der Anwendung und der Verbindungen zu ihnen. Es wird z.B. ausgegeben, ob der Kommunikationspartner zur Zeit mit der Anwendung verbunden ist und wie lange diese Verbindung bereits besteht, ob er zur Zeit gesperrt ist oder nicht, wieviele Nachrichten auf der Verbindung ausgetauscht wurden, ob der automatische Verbindungsaufbau generiert ist.
- Anzahl der Objekte eines bestimmten Objekttyps, die maximal in der Konfiguration der Anwendung enthalten sein können.
- Anzahl der Objekte, die noch dynamisch in die Konfiguration aufgenommen werden können.

Welche Daten im einzelnen zurückgeliefert werden, ist im Abschnitt „Datenstrukturen zur Informationsübergabe" für die Abfrage mit KC_GET_OBJECT und im Abschnitt "Ausgabe von KDCINF" für die Abfrage mit dem Administrationskommando KDCINF beschrieben.

Bei der Informationsabfrage können Sie Selektionskriterien angeben, d.h. Sie können Informationen zu Objekten anfordern, die bestimmte Eigenschaften haben, z.B.:

- alle LU6.1-Verbindungen, die zur Zeit aufgebaut sind,
- die Association-Id aller Associations, die zur Zeit zu einer OSI TP-Partner-Anwendung aufgebaut sind,
- alle Clients und Drucker, die derzeit mit der Anwendung verbunden sind.
- alle Benutzer, die derzeit mit der Anwendung verbunden sind
- alle LTERMs eines Verbindungsbündels oder alle (OSI-)LPAPs eines LPAP-Bündels

3.2 Performance-Kontrolle

openUTM bietet Ihnen zahlreiche Funktionen an, mit denen Sie sich über die Auslastung der Anwendung informieren, Engpässe diagnostizieren und Maßnahmen zur Performance-Verbesserung einleiten können.

Gründe für Performance-Engpässe können z.B. sein:

- Zunahme von Service-Aufrufen in Spitzenzeiten.
- Zu viele Benutzer/Clients arbeiten gleichzeitig mit der Anwendung.
- Die Prozesse, die der Anwendung zur Verfügung stehen, sind lange von Aufträgen belegt, weil auf von anderen Prozessen gesperrte Betriebsmittel gewartet werden muss.
- Die Bearbeitung vieler Asynchron-Aufträge beeinträchtigt den Dialog-Betrieb.
- Es laufen viele lang laufende Teilprogramme (Langläufer) gleichzeitig, z.B. Teilprogramme, die Datenbestände nach bestimmten Informationen durchsuchen.
- Es laufen viele Teilprogramme gleichzeitig, die blockierende Aufrufe enthalten, z.B. den KDCS-Aufruf PGWT oder den XATMI-Aufruf *tpcall*. Während des Wartens belegt jedes dieser Teilprogramme einen Prozess der Anwendung exklusiv.
- Bei der verteilten Verarbeitung über OSI TP oder LU6.1 wird lange auf die Belegung einer Association oder Session gewartet.
- Häufige I/O-Zugriffe auf den Pagepool.
  Häufige Lesezugriffe können ein Indiz dafür sein, dass der Cache der UTM-Anwendung zu klein generiert ist.
- Engpässe auf den Verbindungen zu Kommunikationspartnern der Anwendung.

3.2.1 Informationen über die Auslastung der Anwendung

Anhand der Daten über die momentane und maximale Auslastung der Anwendung sowie die Auslastung einzelner Objekte, die die Informationsfunktionen von openUTM liefern, können Sie drohende Engpässe erkennen und frühzeitig Maßnahmen ergreifen, um Engpässe zu vermeiden.

Wichtige Daten für die Performance-Kontrolle können Sie mit folgenden Aufrufen abfragen:

KDCINF STATISTICS oder SYSPARM (allgemeine Daten)
KDCINF *objekttyp* (Abfrage der Daten für einzelne Objekte)
Welche Daten KDCINF STATISTICS zurückliefert, ist "type=STATISTICS" im Abschnitt "Ausgabe von KDCINF" beschrieben.

KC_GET_OBJECT mit *obj_type*=KC_CURR_PAR (allgemeine Daten)
Für die Abfrage Objekt-bezogener Daten geben Sie in *obj_type* den Objekttyp des Objektes an.
Welche Daten bei der Abfrage mit KC_CURR_PAR zurückgeliefert werden ist im Abschnitt "kc_curr_par_str - Aktuelle Werte der Anwendungsparameter" beschrieben.
Objekt-spezifische Daten finden Sie im Abschnitt „Datenstrukturen zur Beschreibung der Objekteigenschaften".

KC_GET_OBJECT mit *obj_type*=KC_CLUSTER_CURR_PAR für Unix-, Linux- und Windows-Systeme
Liefert Daten zur Belegung des Cluster-Pagepools in UTM-Cluster-Anwendungen, siehe Abschnitt "kc_cluster_curr_par_str - Statistikwerte einer UTM-Cluster-Anwendung".

Weisen die obigen Informationsfunktionen auf Leistungsengpässe hin, dann sollten Sie detaillierte Untersuchungen mit Hilfe des UTM-Messmonitors KDCMON durchführen. KDCMON erstellt z.B. Statistiken über die Auslastung der Anwendung, den Ablauf von Anwendungs-Teilprogrammen und Bearbeitungszeiten von Aufträgen. KDCMON können Sie mit Hilfe der Administration im laufenden Betrieb ein- und nach einer gewünschten Messdauer wieder ausschalten. Die Daten können Sie mit dem UTM-Tool KDCEVAL auswerten.

KDCAPPL KDCMON

KC_MODIFY_OBJECT mit *obj_type*=KC_DIAG_AND_ACCOUNT_PAR

Der Messmonitor KDCMON und das Tool KDCEVAL sind im openUTM-Handbuch „Einsatz von UTM-Anwendungen “ beschrieben. Dort finden Sie auch Interpretationshilfen zu den Statistiken von KDCMON und mögliche Maßnahmen bei Leistungsengpässen.

Für Leistungsmessungen steht der Software-Monitor openSM2 zur Verfügung. openSM2 liefert statistische Daten über die Leistung des gesamten Anwendungsprogramms und die Auslastung der Betriebsmittel. Per Administration können Sie die Datenlieferung an openSM2 ein- und ausschalten. Zu openSM2 siehe auch openUTM-Handbuch „Einsatz von UTM-Anwendungen“.

KDCAPPL SM2

KC_MODIFY_OBJECT mit *obj_type*=KC_MAX_PAR

3.2.2 Diagnose von Fehlern und Engpässen

openUTM stellt folgende Funktionen zur Verfügung, die Sie bei der Diagnose von Leistungsengpässen und Fehlverhalten der Anwendung unterstützen:

- Die maximale Auslastung einer Anwendung in einem bestimmten Zeitraum prüfen.
- Protokollierung von Ereignissen in Form von UTM-Meldungen in der SYSLOG
- Zur Diagnose von Engpässen und Fehlern auf Verbindungen zu Kommunikationspartnern können Sie den UTM-BCAM-Trace oder den OSS-Trace einschalten. Der UTM-BCAM-Trace kann für alle Verbindungen eingeschaltet werden, nur für bestimmte Benutzer, oder nur für Verbindungen zu bestimmten Partner-Anwendungen und Clients.
- Zur Diagnose von Fehlern in Teilprogrammen, die die X/Open-Schnittstellen CPI-C, TX oder XATMI verwenden, können Sie den CPI-C-Trace, den TX-Trace oder den XATMI-Trace einschalten.
- Zur Diagnose von Fehlern an der Programmschnittstelle zur Administration (KDCADMI) können Sie den ADMI-Trace einschalten.
- Testmodus einschalten. Der Testmodus dient zur Erzeugung von Diagnoseunterlagen bei Fehlern im UTM-Systemcode. Da sich der Testmodus negativ auf die Performance einer UTM-Anwendung auswirkt, sollten Sie den Testmodus nur einschalten, nachdem Sie vom Systemdienst dazu aufgefordert wurden. Im Testmodus werden zusätzliche UTM-interne Plausibilitätsprüfungen durchgeführt und interne Trace-Informationen aufgezeichnet.
- Einen Diagnose-Dump anfordern, ohne dass dafür der Anwendungslauf unterbrochen werden muss. Dabei können Sie per Kommando oder Programmschnittstelle
  - sofort einen allgemeinen Diagnose-Dump anfordern. Dieser hat die Kennzeichnung DIAGDP.
  - oder einen Dump anfordern, sobald ein bestimmtes Ereignis (Meldung, KDCS-Returncode, Signon-Returncode) von openUTM erzeugt wird. Die Kennzeichnung des Dumps ist abhängig vom Ereignis. Zuvor müssen Sie den Testmodus einschalten, da der Dump nur bei eingeschalteten Testmodus geschrieben wird.

    KDCDIAG

    KC_MODIFY_OBJECT mit *obj_type*=KC_DIAG_AND_ACCOUNT_PAR

3.2.3 Mögliche Maßnahmen

Im Folgenden finden Sie einige Maßnahmen, die Sie ergreifen können, um Performance-Engpässe zu vermeiden oder bestehende Engpässe zu beseitigen.

Gesamtprozesszahl der Anwendung heraufsetzen

Kommt es insbesondere im Dialog-Betrieb bei der Bearbeitung von Aufträgen zu großen Wartezeiten, dann können Sie die Anzahl der Prozesse, in denen das Anwendungsprogramm abläuft, erhöhen.

Dies ist insbesondere dann sinnvoll, wenn die aktuelle Auslastung der Anwendung auf über 80 Prozent ansteigt und gleichzeitig noch genügend Systemressourcen auf dem Rechner frei sind (Speicherplatz, CPU-Leistung). Nach ausreichender Erhöhung der Gesamtprozesszahl sollte sich dieser Wert wieder verringern.

Die erlaubte Maximalzahl der Prozesse wird bei der KDCDEF-Generierung in MAX TASKS festgelegt. Diese Maximalzahl kann administrativ nicht erhöht werden. Ist die momentan eingestellte Prozesszahl jedoch kleiner als diese Maximalzahl, dann können Sie weitere Prozesse für die Anwendung starten.

KDCINF SYSPARM:
Aktuelle Maximalzahl der Prozesse, maximal erlaubte Prozesszahl abfragen.

KDCAPPL TASKS: Neue Prozesszahl festlegen

KC_GET_OBJECT mit *obj_type*=KC_TASKS_PAR:
Abfragen der erlaubten Maximalzahl und der derzeit eingestellten Prozesszahl
KC_MODIFY_OBJECT mit *obj_type*=KC_TASKS_PAR: Ändern der Prozesszahl

Gesamtprozesszahl der Anwendung herabsetzen

Wegen möglicher Lastschwankungen ist es normalerweise nicht sinnvoll, die Gesamtprozesszahl herabzusetzen, wenn die Anwendung zeitweise nicht voll ausgelastet ist.

Die Gesamtprozesszahl sollte nur dann reduziert werden, wenn der Rechner insgesamt in einen Engpass gerät, der eine Verschlechterung des Durchsatzes und/oder der Antwortzeiten der Anwendung bewirkt.

Wenn Sie die Gesamtprozesszahl herabsetzen, müssen Sie Folgendes beachten:

- Wird die Gesamtprozesszahl so weit herabgesetzt, dass sie kleiner ist als die aktuell eingestellte Maximalzahl der Prozesse, die gleichzeitig für die Asynchron-Verarbeitung verwendet werden können (im Folgenden ASYNTASKS genannt), dann setzt openUTM den Wert von ASYNTASKS auf die angegebene Gesamtprozesszahl zurück. Bei späteren Änderungen der Gesamtprozesszahl passt openUTM den Wert von ASYNTASKS automatisch an, bis der Wert erreicht ist, der zuvor durch die Administration oder im Startparameter ASYNTASKS eingestellt wurde.

  Analoges gilt für die Maximalzahl der Prozesse, in denen gleichzeitig Teilprogramme mit blockierende Aufrufen ablaufen dürfen (TASKS-IN-PGWT). Beachten Sie jedoch, dass die Gesamtprozessanzahl mindestens 2 betragen muss, wenn ein Transaktionscode oder eine TAC-Klasse mit PGWT=YES generiert ist oder wenn es sich um eine UTM-Cluster-Anwendung handelt.
- Ist für eine Dialog-TAC-Klasse der Wert von TASKS-FREE größer als die aktuelle Gesamtprozesszahl, dann bearbeitet weiterhin ein Prozess die Aufträge an diese TAC-Klasse.
- Wird die Auftragsbearbeitung in der Anwendung über Prioritäten gesteuert (es ist TAC-PRIORITIES generiert) und ist der Wert für FREE-DIAL-TASKS größer als die aktuelle Gesamtprozesszahl, dann bearbeitet weiterhin ein Prozess Aufträge an die Dialog-TAC-Klassen.

Damit der Dialog-Betrieb nach dem Herabsetzen der Gesamtprozesszahl nicht durch langlaufende Asynchron-Vorgänge oder durch Programme mit blockierenden Aufrufen beeinträchtigt wird, sollten Sie die Werte von ASYNTASKS und TASKS-IN-PGWT anpassen, d.h. ebenfalls herabsetzen.

Anzahl der Prozesse herabsetzen, die für Asynchron-Verarbeitung und Bearbeitung von Teilprogrammen mit blockierenden Aufrufen zur Verfügung steht

Wird der Dialog-Betrieb der Anwendung durch zeitintensive Asynchron-Verarbeitung verzögert (Dialog-Aufträge warten, weil zu viele Prozesse gleichzeitig Asynchron-Aufträge bearbeiten), dann können Sie die Maximalzahl der Prozesse (ASYNTASKS) reduzieren, die gleichzeitig für die Asynchron-Verarbeitung verwendet werden können. Damit bleiben mehr Prozesse für die Synchron-Verarbeitung frei. Die Anzahl der Prozesse ASYNTASKS ist durch den in MAX ASYNTASKS generierten Maximalwert beschränkt.

Sie können ASYNTASKS zeitweise auf 0 setzen. Dabei sollten Sie jedoch beachten, dass alle Asynchron-Aufträge im Pagepool zwischengespeichert werden. Ist der Pagepool nicht groß genug dimensioniert, kann es zu Pagepool-Engpässen kommen.

Beim Herabsetzen von ASYNTASKS müssen Sie, wenn in Ihrer Anwendung die Steuerung der Aufträge durch Prozessbeschränkung für die einzelnen TAC-Klassen gesteuert wird (Generierung ohne TAC-PRIORITIES-Anweisung), auch Folgendes beachten:

Existiert eine Asynchron-TAC-Klasse, für die der aktuell eingestellte Wert von TASKS-FREE größer als oder gleich ASYNTASKS ist, dann wird diese TAC-Klasse gesperrt, d.h. es werden keine Aufträge für diese TAC-Klasse mehr bearbeitet. TASKS-FREE ist in diesem Fall die Anzahl der Prozesse, die mindestens für die Bearbeitung von Aufträgen an andere Asynchron-TAC-Klassen freigehalten werden soll.

Zur Kontrolle sollten Sie nach dem Herabsetzen von ASYNTASKS Informationen über die TAC-Klassen anfordern.

Analoges gilt für die Maximalzahl der Prozesse, in denen gleichzeitig Teilprogramme mit blockierende Aufrufen ablaufen dürfen (TASKS-IN-PGWT). Beachten Sie jedoch, dass Sie - im Gegensatz zu ASYNTASKS - diese Anzahl *nicht* auf 0 setzen können, wenn derartige Prozesse generiert sind.

KDCINF SYSPARM:
Aktuelle Einstellungen anzeigen
KDCAPPL ASYNTASKS / TASKS-IN-PGWT: Prozesszahl ändern

KC_GET_OBJECT mit *obj_type*=KC_TASKS_PAR:
Generierte Maximalzahl und die aktuell eingestellte Prozesszahl ermitteln
KC_MODIFY_OBJECT mit *obj_type*=KC_TASKS_PAR: Prozesszahl ändern

In Anwendungen ohne TAC-PRIORITIES-Anweisung: Prozesszahlen für die einzelnen TAC-Klassen ändern

Ist Ihre Anwendung mit TAC-Klassen ohne die Prioritätensteuerung generiert, dann können Sie die Anzahl der Prozesse, die maximal Aufträge einer TAC-Klasse bearbeiten, für jede TAC-Klasse gesondert festlegen und nach Bedarf ändern.

Beim Eintragen der Transaktionscodes geben Sie an, zu welcher TAC-Klasse ein Transaktionscode gehören soll. Sie können also die Transaktionscodes, die zu langlaufenden Teilprogrammen gehören, in einer oder mehreren TAC-Klassen zusammenfassen. Den Anteil der Prozesse der Anwendung, der gleichzeitig Aufträge für diese TAC-Klassen bearbeiten darf, können Sie dann je nach Auslastung der Anwendung einstellen. Bei Dialog-TAC-Klassen muss mindestens ein Prozess Aufträge der TAC-Klasse bearbeiten dürfen. Bei Asynchron-TAC-Klassen kann die Zahl auf 0 herabgesetzt werden.

Insbesondere sollten Sie die Dialog-TACs der Teilprogramme, die blockierende Aufrufe enthalten (z.B. KDCS-Aufruf PGWT, oder XATMI-Aufruf *tpcall*), in eine TAC-Klasse (mit PGWT=YES) zusammenfassen.

Nach einem blockierenden Aufruf wartet das Teilprogramm, bis die für den weiteren Programmablauf benötigten Daten eingetroffen sind. Für diese Zeit belegt das Teilprogramm bzw. der zugehörige Transaktionscode einen Prozess der Anwendung exklusiv.

Laufen mehrere derartige Teilprogramme gleichzeitig, dann kann es passieren, dass andere Aufträge in der Warteschlange stehen bleiben, weil keine Prozesse für ihre Bearbeitung verfügbar sind. Die Performance der Anwendung wird dadurch stark beeinträchtigt. Die Wartezeit nach einem blockierenden Aufruf kann auch mit dem Timer PGWTTIME begrenzt werden (siehe unten).

KDCINF TACCLASS:
Aktuelle Einstellung ermitteln
KDCTCL: Prozesszahl ändern

KC_GET_OBJECT mit *obj_type*=KC_TACCLASS:
Aktuelle Einstellung ermitteln
KC_MODIFY_OBJECT mit *obj_type*=KC_TACCLASS: Prozesszahl ändern

Einstellung der Timer ändern

Um zu verhindern, dass Prozesse der Anwendung unnötig lange belegt sind, weil sie auf das Freiwerden von Betriebsmitteln oder auf den Aufbau von Verbindungen und Sessions warten, sind Timer definiert. Die Timer überwachen diese Wartezeiten und setzen die wartende Transaktion nach Ablauf der angegebenen Zeit zurück. Die Timer werden bei der KDCDEF-Generierung festgelegt und können im laufenden Betrieb angepasst werden.

In openUTM sind u.a. Timer für folgende Wartezeiten definiert:

- Wartezeit nach einem blockierenden Aufruf (*pgwttime*)
  Der Timer überwacht die Wartezeit, die ein Teilprogramm nach dem Absetzen eines blockierenden Aufrufs maximal auf die Rückkehr ins Teilprogramm wartet.
- Zeit, die innerhalb einer Transaktion maximal auf die Antwort von einem Dialog-Partner gewartet wird (*termwait*...).
- Zeit, die ein Betriebsmittel maximal von einer Transaktion belegt werden darf, und Zeit, die ein Teilprogramm maximal auf das Freiwerden von Betriebsmitteln wartet (*reswait*...).

  Mit den Informationsfunktionen (Parametertyp STATISTICS/KC_CURR_PAR) können Sie z.B. ermitteln, wie oft Teilprogramme auf gesperrte Betriebsmittel warten mussten (relative Angabe).
- Zeit, die maximal auf die Belegung einer Session/Association zur Partner-Anwendung gewartet wird.

> **i** Die Timer sind nur als "Notbremse" für unvorhergesehene Situationen gedacht. Stellen Sie die Timer-Werte daher so ein, dass die Timer beim normalen Ablauf der Anwendung nicht ablaufen. Nur Ausnahmesituationen, wie z.B. ein Programmfehler oder eine ausbleibende Antwort von einer Partner-Anwendung, sollten zu einem Timeout führen.

Sind die Timer *pgwttime* oder *reswait* zu groß eingestellt, dann können die einzelnen Prozesse der Anwendung gerade in Engpass-Situationen zu lange von Teilprogrammen belegt sein, die entweder Betriebsmittel zu lange sperren (Langläufer) oder zu lange auf das Freiwerden benötigter Betriebsmittel warten. Sind die Timer zu klein eingestellt, wird die Performance durch häufiges Rücksetzen von Transaktionen belastet.

KDCINF SYSPARM oder STATISTICS:
Aktuelle Timer-Einstellungen ermitteln, Informationen über aktuelle Wartezeiten.

KDCAPPL: Timer-Einstellung ändern

KC_GET_OBJECT mit *obj_type*=KC_TIMER_PAR / KC_CURR_PAR:
Aktuelle Timer-Einstellungen ermitteln, Informationen über aktuelle Wartezeiten
KC_MODIFY_OBJECT mit *obj_type*=KC_TIMER_PAR: Timer-Einstellung ändern

Die Anzahl der angemeldeten Benutzer/Clients begrenzen

Sie können im laufenden Betrieb die Anzahl der Benutzer/Clients beeinflussen, die sich gleichzeitig an die Anwendung anschließen und Services von der Anwendung anfordern können. Dazu stehen Ihnen folgende Möglichkeiten zur Verfügung:

- Sie können die Gesamtzahl der Benutzer/Clients, die gleichzeitig bei der Anwendung angemeldet sein können, begrenzen.
- Sie können die Anzahl der Clients, die sich gleichzeitig über die einzelnen LTERM-Pools anschließen können, beschränken. Dazu sperren Sie einen Teil der LTERM-Partner des Pools.
- Sie können einzelne Clients/LTERM-Partner/Benutzer sperren.
- Sie können LTERM-Pools vollständig sperren. Über einen gesperrten LTERM-Pool kann sich kein Benutzer/Client mehr an die Anwendung anschließen.
- Nur auf BS2000-Systemen: Sie erlauben nur eine geringe Anzahl paralleler Sessions zu einem Multiplexanschluss.

KDCAPPL MAX-CONN-USERS: Gesamtzahl der Benutzer/Clients
KDCPOOL: Anzahl der Pool-LTERM-Partner festlegen / LTERM-Pool sperren

KC_MODIFY_OBJECT
obj_type=KC_MAX_PAR: Gesamtzahl der Benutzer/Clients festlegen
obj_type=KC_TPOOL:
Anzahl der Pool-LTERM-Partner festlegen / LTERM-Pool sperren
obj_type=KC_PTERM: Client / Drucker sperren
obj_type=KC_LTERM: LTERM-Partner sperren
obj_type=KC_USER: Benutzer sperren

Services sperren

Sie können z.B. langlaufende Services zeitweise sperren, indem Sie den zugehörigen Transaktionscode sperren (Status OFF). Aufträge für gesperrte Transaktionscodes werden nicht mehr angenommen. Bei gesperrten Asynchron-TACs werden keine Aufträge mehr in die Message Queue geschrieben.

Sie können einen Transaktionscode entweder nur in seiner Eigenschaft als Vorgangs-TAC sperren oder Sie können ihn als Vorgangs- und als Folge-TAC sperren (vollständiges Sperren: Status HALT).

Asynchron-Services können Sie auch mit dem Status KEEP sperren. KEEP bedeutet, dass für den Asynchron-TAC zwar Aufträge entgegengenommen, aber noch nicht bearbeitet werden. Die Bearbeitung der Aufträge kann dann zu einem Zeitpunkt erfolgen, zu dem die Anwendung weniger ausgelastet ist, z.B. nachts.

KDCTAC

KC_MODIFY_OBJECT *obj_type* =KC_TAC

Engpässe auf Verbindungen zu Partner-Anwendungen verhindern bzw. beheben

Kommt es bei der Kommunikation mit LU6.1- oder OSI TP-Partner-Anwendungen zu Engpässen, können Sie folgende Aktionen durchführen:

- Weitere Transportverbindungen zu einer LU6.1-Partner-Anwendung aufbauen. Voraussetzung ist, dass Sie für die Kommunikation zu der Partner-Anwendung mehrere parallele Verbindungen erzeugt oder generiert haben und dass noch nicht alle erzeugten oder generierten Verbindungen aufgebaut sind.
- Die Anzahl der parallelen logischen Verbindungen zu einer OSI TP-Partner-Anwendung erhöhen. Die maximal mögliche Anzahl der parallelen Verbindungen wird bei der Generierung in der OSI-LPAP-Anweisung festgelegt.
- Den Timer (Accesswait) für die Zeit anpassen, die nach dem Anfordern eines fernen Services auf das Freiwerden bzw. den Aufbau einer Session oder Association zur Partner-Anwendung gewartet wird. Diesen Timer können Sie LTAC-spezifisch einstellen. Wird der Timer für einen Asynchron-LTAC auf 0 gesetzt, dann werden Asynchron-Aufträge für diesen LTAC auch nicht in die lokale Message Queue der Partner-Anwendung eingereiht.
- Den Timer (Replywait) anpassen, der das Warten auf die Antwort von der Partner-Anwendung überwacht. Dieser Timer wird ebenfalls LTAC-spezifisch eingestellt.
- Die Einstellung des Leerlauf-Timers anpassen (Idletime). Dieser Timer gibt die Zeit an, die sich eine Session oder Association ungenutzt bleiben darf, bevor openUTM die Verbindung zur Partner-Anwendung abbaut. Ist der Timer zu groß eingestellt, werden unnötig Ressourcen an Verbindungen gebunden, die nicht benötigt werden. Ist der Timer zu klein eingestellt, dann werden zu viele Ressourcen verbraucht, um die Verbindung immer wieder aufzubauen. Der Timer wird Partner-spezifisch eingestellt.

KDCLPAP / KDCLSES: Verbindungen aufbauen, Idletime anpassen
KDCLTAC: Accesswait und Replywait ändern

KC_CREATE_OBJECT *obj_type*=KC_CON/KC_LSES:
Verbindungen und Sessions erzeugen

KC_MODIFY_OBJECT *obj_type*=KC_LPAP/KC_OSI_LPAP
/KC_LSES:
Verbindungen aufbauen, Idletime anpassen
obj_type=KC_LTAC: Accesswait und Replywait ändern

Hinweis für Unix-, Linux- und Windows-Systeme:

Werden in Ihrer Anwendung eine sehr große Anzahl von Verbindungen über denselben BCAMAPPL-Namen oder Zugriffspunkt Ihrer Anwendung abgewickelt, dann kann es zu Engpässen kommen, da Prozesse dann an Systemgrenzen stoßen können (z.B. maximale Anzahl der Filedeskriptoren). Bei der nächsten KDCDEF-Generierung sollten Sie dann mehr BCAMAPPL-Namen und Zugriffspunkte generieren.

Datenkomprimierung ein- oder ausschalten

Falls häufig GSSBs, LSSBs, ULS, TLS oder KB-Programmbereiche in einer Länge größer einer UTM-Seite gelesen oder geschrieben werden, sollten Sie prüfen, ob das Einschalten der Datenkomprimierung die Performance der UTM-Anwendung verbessert.

Ob sich die Datenkomprimierung lohnt, können Sie bei eingeschalteter Datenkomprimierung wie folgt prüfen:

KDCINF STAT, Feld *AVG COMPRESS PAGES SAVED*

KC_GET_OBJECT mit *obj_type* =KC_CURR_PAR, Feld *avg_saved_pgs_by_compr*

3.3 Pagepool-Engpass vermeiden

Der Inhalt und die Rolle des Pagepools hängt davon ab, ob es sich um eine stand-alone-Anwendung (siehe unten) oder eine UTM-Cluster-Anwendung (siehe Kapitel "Pagepools einer UTM-Cluster-Anwendung") handelt.

3.3.1 Pagepool einer stand-alone-Anwendung

Im Pagepool einer stand-alone-Anwendung werden Benutzerdaten gespeichert, die während des Anwendungslaufs erzeugt werden. Das sind neben den UTM-Speicherbereichen und den Vorgangsdaten u.a.:

- Die Message Queues der Asynchron-TACs, der LTERM-, LPAP- und OSI-LPAP-Partner sowie Benutzer-, TAC- und temporäre Queues d.h. Aufträge an lokale Services und Kommunikationspartner, service-gesteuerte Queues sowie Druckaufträge an die Drucker der Anwendung, die noch nicht bearbeitet sind.
- Zwischengespeicherte Dialog- oder Asynchron-Aufträge an Transaktionscodes von TAC-Klassen, die als Folge der TAC-Klassen-Steuerung unterbrochen werden.

Die Pagepoolgröße wird bei der KDCDEF-Generierung festgelegt und kann im laufenden Betrieb nicht verändert werden.

Bei laufender Anwendung muss verhindert werden, dass der Pagepool vollständig belegt wird. Dafür werden bei der KDCDEF-Generierung zwei Warnstufen (Pagepool-Belegung in %) definiert. Erreicht die Pagepool-Belegung eine dieser Warnstufen, dann erzeugt openUTM die Meldung K041. Wird für diese Meldung das Meldungsziel MSGTAC definiert, können Sie in einer MSGTAC-Routine auf dieses Ereignis reagieren. Wird die 2. Warnstufe (Standardeinstellung 95%) erreicht, dann werden keine Asynchron-Aufträge mehr in die Message Queues und keine Benutzer-Protokollsätze (LPUT-Aufträge) mehr in die Benutzerprotokolldatei geschrieben. Asynchron-Aufträge und LPUT-Aufrufe werden dann abgewiesen.

Deshalb sollten Sie bei Erreichen der 1. Warnstufe Maßnahmen ergreifen, um Speicherplatz im Pagepool freizugeben. Im Betrieb der Anwendung können Sie sich über die momentane Belegung des Pagepools informieren.

KDCINF STATISTICS
KDCINF PAGEPOOL

KC_GET_OBJECT mit *obj_type*=KC_CURR_PAR
KC_GET_OBJECT mit *obj_type*=KC_PAGEPOOL

Kommt es allerdings häufig zu Pagepool-Engpässen, dann ist der Pagepool zu klein dimensioniert. Sie sollten dann eine Neugenerierung der Anwendung durchführen und den Pagepool vergrößern.

Im Folgenden ist beschrieben, wie Sie die Message Queues und zwischengespeicherte Dialog-Aufträge abbauen können, um die Überbelegung des Pagepools abzubauen.

Message Queues abbauen

Folgende Maßnahmen können Sie ergreifen, um Message Queues abzubauen:

- Drucker-Queues können Sie abbauen, indem Sie Verbindung zu allen Druckern aufbauen, für die Druckaufträge vorliegen. Die Druckaufträge werden dann sofort bearbeitet, auch wenn für einen Drucker ein Schwellwert generiert (*plev*) und dieser noch nicht erreicht ist.
- Die Verbindungen zu TS-Anwendungen und Partner-Anwendungen aufbauen, für die Asynchron-Aufträge im Pagepool zwischengespeichert sind. Sind die Kommunikationspartner gesperrt, müssen sie zuvor freigegeben werden.
- Die Anzahl der Prozesse heraufsetzen, die gleichzeitig für die Asynchron-Verarbeitung verwendet werden können.
- Die Prozesszahl heraufsetzen, die maximal gleichzeitig zur Bearbeitung von Aufträgen an eine bestimmte TAC-Klasse verwendet werden kann (in Anwendungen ohne Prioritätensteuerung).

- Asynchron-Transaktionscodes und TAC-Queues, die mit Status KEEP gesperrt bzw. blockiert sind, entweder entsperren (Status ON) oder mit Status OFF sperren. Status KEEP bedeutet, dass Aufträge an den Transaktionscode bzw. an die Queue zwar angenommen, aber nicht bearbeitet werden. Status OFF bedeutet, dass keine weiteren Aufträge angenommen, wartende Aufträge jedoch abgearbeitet werden.
- Die Asynchron-Aufträge löschen, die in den Message Queues dynamisch gelöschter LTERM-Partner und Asynchron-TACs stehen.
- Ältere Nachrichten aus service-gesteuerten Queues löschen, wenn diese voraussichtlich nicht mehr gelesen werden.
- Nachrichten der Dead Letter Queue wieder einem neuen Ziel zuordnen, damit sie verarbeitet werden können.

KDCINF STATISTICS:
Gesamtzahl aller im Pagepool zwischengespeicherten Nachrichten abfragen

KDCINF LTERM / LPAP / OSI-LPAP / TAC:
Belegung der Message Queues einzelner Objekte abfragen

KDCINF PAGEPOOL:
Pagepoolseitenbelegung nach Typen aufgeschlüsselt abfragen

KDCAPPL SPOOLOUT: Drucker-Queues abbauen

KDCLTERM bzw. KDCLPAP: Verbindung zu Kommunikationspartner aufbauen

KDCAPPL ASYNTASKS: Ändern der Prozesszahl

KDCTAC STATUS: Status eines Transaktionscodes ändern

KDCTCL: Ändern der Prozesszahl einer TAC-Klasse

KC_GET_OBJECT
mit *obj_type*=KC_CURR_PAR: Gesamtzahl aller im Pagepool zwischengespeicherten Nachrichten abfragen
mit *obj_type*=KC_LTERM / KC_LPAP / KC_OSI-LPAP / KC_TAC: Belegung der Message Queues einzelner Objekte abfragen
mit *obj_type*=KC_PAGEPOOL:
Pagepoolseitenbelegung nach Typen aufgeschlüsselt abfragen

KC_SPOOLOUT: Drucker-Queues abbauen

KC_MODIFY_OBJECT
mit *obj_type*=KC_LTERM / KC_LPAP/KC_OSI_LPAP: Verbindungen aufbauen
mit *obj_type*=KC_TASKS_PAR: Prozesszahl ASYNTASKS ändern
mit *obj_type*=KC_TAC: Status eines Transaktionscodes bzw. einer TAC-Queue ändern
mit *obj_type*=KC_TACCLASS: Ändern der Prozesszahl einer TAC-Klasse

DADM (KDCS-Aufruf): Löschen von Aufträgen und Verschieben von Nachrichten aus der Dead Letter Queue

In Anwendungen ohne TAC-PRIORITIES-Anweisung: Auftrags-Queues der TAC-Klassen abbauen

Wieviele Aufträge für eine bestimmte TAC-Klasse im Pagepool zwischengespeichert sind, können Sie mit den Informationsfunktionen abfragen. In den Informationen, die openUTM zu einer TAC-Klasse ausgibt, ist die Anzahl der im Pagepool zwischengespeicherten Nachrichten enthalten.

Um diese Queues abzubauen, können Sie die maximale Anzahl der Prozesse heraufsetzen, die gleichzeitig Aufträge für diese TAC-Klasse bearbeiten können.

KDCINF TACCLASS: Anzahl der zwischengespeicherten Dialog-Aufträge abfragen
KDCTCL: Prozesszahl ändern

KC_GET_OBJECT mit *obj_type*=KC_TACCLASS:
Anzahl der zwischengespeicherten Dialog-Aufträge abfragen

KC_MODIFY_OBJECT mit *obj_type*=KC_TACCLASS: Prozesszahl ändern

Datenkomprimierung ein- oder ausschalten

Wenn viele Pagepoolseiten für GSSBs, LSSBs, TLS oder ULS belegt sind (KDCINF PAGEPOOL bzw. KC_GET_OBJECT mit *obj_type*=KC_PAGEPOOL), sollten Sie prüfen, ob das Einschalten der Datenkomprimierung die Anzahl der belegten Seiten eventuell reduziert.

Ob sich die Datenkomprimierung lohnt, können Sie bei eingeschalteter Datenkomprimierung wie folgt prüfen:

KDCINF STAT, Feld *AVG COMPRESS PAGES SAVED*

KC_GET_OBJECT mit *obj_type*=KC_CURR_PAR, Feld *avg_saved_pgs_by_compr*

3.3.2 Pagepools einer UTM-Cluster-Anwendung

In einer UTM-Cluster-Anwendung besitzt jede Knoten-Anwendung einen eigenen Pagepool für Knoten-lokale Daten. Zusätzlich gibt es den für alle Knoten-Anwendungen gemeinsamen Cluster-Pagepool für Cluster-weit gültige Daten. Damit ergeben sich im Vergleich zu einer stand-alone-Anwendung einige Besonderheiten:

- Knoten-lokale Daten werden ausschließlich im Pagepool der betreffenden Knoten-Anwendung gehalten. Knoten-lokale Daten sind z.B. die TLS-Bereiche, Message Queues sowie zwischengespeicherte Dialog- oder Asynchron-Aufträge an Transaktionscodes von TAC-Klassen, die als Folge der TAC-Klassen-Steuerung unterbrochen werden.
- Cluster-weit gültige Daten werden im Cluster-Pagepool gehalten. Cluster-weit gültige Daten sind GSSB, ULS sowie Cluster-weit gültige Vorgangsdaten.

Eigenschaften des Cluster-Pagepools

Der Cluster-Pagepool gehört zu den UTM-Cluster-Dateien und besteht aus einer Verwaltungsdatei und einer oder mehreren Dateien, die die Anwenderdaten enthalten. Bei der Generierung mit KDCDEF werden festgelegt:

- Die Größe der Cluster-Pagepool-Datei(en)
- Die Anzahl der Cluster-Pagepool-Dateien
- Eine Warnstufe für den Cluster-Pagepool

Die Meldung für die Über- oder Unterschreitung der Warnstufe wird immer durch die Knoten-Anwendung ausgegeben, die die Zustandsänderung ausgelöst hat.

Per Administration sind folgende Aktionen möglich:

- Sie können die aktuelle Belegung des Cluster-Pagepools ermitteln und die Statistikwerte zurücksetzen, z.B. über WinAdmin, WebAdmin oder die Programmschnittstelle KDCADMI.

  KC_GET_OBJECT und KC_MODIFY_OBJECT mit *obj_type*=KC_CLUSTER_CURR_PAR

- Sie können die Dateien des Cluster-Pagepools vergrößern, ohne die UTM-Cluster-Anwendung zu beenden.

  openUTM-Handbuch „Einsatz von UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen“, Stichwort „Vergrößerung des Cluster-Pagepools“ im Kapitel "Änderungsgenerierung im Cluster".

3.4 Anwendungsprogramm austauschen

Sie können mit Hilfe der Administrationsfunktionen von openUTM das gesamte Anwendungsprogramm oder Teile des Anwendungsprogramms (einzelne Lademodule oder Shared Objects) austauschen, ohne die Anwendung beenden zu müssen.

Voraussetzung für den Austausch einzelner Teile des Anwendungsprogramms ist, dass das UTM-Anwendungsprogramm mit Lademodulen (auf BS2000-Systemen), Shared Objects (auf Unix- und Linux-Systemen) bzw. DLLs (auf Windows-Systemen) generiert wurde.

Nähere Informationen zum Programmaustausch und den Voraussetzungen für den Programmaustausch finden Sie im openUTM-Handbuch „Einsatz von UTM-Anwendungen“.

KDCAPPL PROGRAM: Austausch des gesamten Anwendungsprogramms
KDCPROG: Austausch einzelner Lademodule, Shared Objects oder DLLs.

KC_CHANGE_APPLICATION: Austausch des gesamten Anwendungsprogramms
KC_MODIFY_OBJECTS mit *obj_type*= KC_LOAD_MODULE: Austausch einzelner Lademodule, Shared Objects oder DLLs.

Besonderheiten auf BS2000-Systemen

Sie müssen beim Austausch von Lademodulen, die in einem Common Memory Pool liegen, wie folgt vorgehen:

1. Sie merken die Lademodule vor, die ausgetauscht werden sollen. Dazu rufen Sie für diese Lademodule KC_MODIFY_OBJECT mit *obj_type*= KC_LOAD_MODULE auf und geben an, welche Version beim folgenden Austausch geladen werden soll. Alternativ können Sie das Kommando KDCPROG verwenden.
2. Um die vorgemerkten Lademodule auszutauschen, muss das gesamte Anwendungsprogramm in den einzelnen Prozessen beendet und neu geladen werden. Dazu rufen Sie KC_CHANGE_APPLICATION auf oder verwenden das Kommando KDCAPPL.

3.5 Clients und Drucker

Für die Clients und Drucker einer UTM-Anwendung können Sie die im Folgenden beschriebenen Aktionen durchführen.

> ¡ Drucker werden von openUTM auf Windows-Systemen nicht unterstützt.

Logische Eigenschaften eines Terminals auf ein anderes Terminal übertragen

Ist z.B. ein Terminal defekt oder will ein Terminal-Benutzer in Zukunft von einem anderen Terminal aus arbeiten, dann können Sie in stand-alone UTM-Anwendungen die logischen Eigenschaften eines Terminals auf ein anderes Terminal übertragen. Dazu ordnen Sie den LTERM-Partner des einen Terminals einem anderen Terminal (gleichen Typs) zu. Damit übertragen Sie z.B. folgende Eigenschaften auf das neue Terminal:

- Wiederanlaufinformationen
- Zugriffsrechte (Keyset)
- Zugriffsschutz (Access-List oder Lockcode)
- Message Queue mit Asynchron-Nachrichten
- Benutzerkennung für das automatische KDCSIGN, sofern definiert
- Sprachumgebung, sofern definiert
- Startformat, sofern definiert
- Schwellwert *qlev* für die Message Queue, sofern definiert

KDCSWTCH

KC_MODIFY_OBJECT mit *obj_type*=KC_PTERM

Message Queue eines Druckers einem anderen Drucker zuordnen

In stand-alone UTM-Anwendungen kann bei Störung eines Druckers die Drucker-Queue einem anderen Drucker (gleichen Typs) zugeordnet werden, der dann die Druckaufträge bearbeitet. Dazu müssen Sie den defekten Drucker sperren und dem LTERM-Partner des Druckers einen anderen Drucker zuordnen.

Neben der Drucker-Queue werden auch die festgelegten logischen Eigenschaften auf den neuen Drucker übertragen, u.a. der Schwellwert *qlev* für die Drucker-Queue und der Wert *plev*. Sobald *plev* Druck-Aufträge in der Drucker-Queue stehen, baut openUTM automatisch die Verbindung zum Drucker auf.

KDCPTERM: Sperren des Druckers
KDCSWTCH: LTERM-Partner einem anderen Drucker zuordnen

KC_MODIFY_OBJECT mit *obj_type*=KC_PTERM

Druckerbündel erzeugen

In stand-alone UTM-Anwendungen können Sie im laufenden Betrieb Drucker der Anwendung zu Druckerbündeln zusammenfassen. Druckerbündel entstehen, wenn Sie dem LTERM-Partner eines Druckers weitere Drucker zuordnen. Die Drucker-Queue, die zu dem LTERM-Partner gehört, wird dann von allen Druckern gemeinsam abgearbeitet, die dem LTERM-Partner zugeordnet werden. Gründe dafür, ein Druckerbündel zu erzeugen, können sein:

- Die Message Queue eines Druckers wird zu groß. Es muss zu lange auf angeforderte Druckausgaben gewartet werden und der Pagepool, in dem die Aufträge zwischengespeichert werden, wird zu stark belastet. Zur Bearbeitung der Druck-Aufträge in der Queue sollten dann mehrere Drucker eingesetzt werden.
- Die beim Eintragen eines Druckers festgelegte Maximalzahl der Druck-Aufträge, die gleichzeitig in der Drucker-Queue zwischengespeichert werden können (*qlev*) ist zu klein dimensioniert, es werden deshalb immer wieder Druckaufträge an diesen Drucker abgewiesen.
- In einer Filiale sind neuerdings weitere Drucker verfügbar. Die Drucker sollen gemeinsam alle Druckaufträge der Filiale bearbeiten. D.h. wird ein Druck-Auftrag gestellt, soll er von dem Drucker bearbeitet werden, der gerade frei ist. Die neuen Drucker können Sie dann dynamisch in die Konfiguration aufnehmen und mit dem zuvor vorhandenen Drucker zu einem Druckerbündel zusammenfassen.

KDCSWTCH

KC_MODIFY_OBJECT mit *obj_type*=KC_PTERM

Drucker/Clients sowie ihre LTERM-Partner sperren

Sie können Clients und Drucker sowie ihre LTERM-Partner sperren. Ein Verbindungsaufbau zu gesperrten Clients bzw. über gesperrte LTERM-Partner ist nicht möglich. Asynchron-Aufträge an gesperrte LTERM-Partner können noch gestellt werden. Sie werden in der Message Queue gespeichert, bis der Schwellwert der Message Queue erreicht ist. Sie werden jedoch erst bearbeitet, wenn die Sperre aufgehoben ist.

KDCLTERM, KDCPTERM

KC_MODIFY_OBJECT mit *obj_type*=KC_PTERM oder KC_LTERM

Verbindungen zu Clients und Druckern

Sie können die Verbindungen zu TS-Anwendungen, Terminals und Druckern der Anwendung bei Bedarf auf- und wieder abbauen.
Für Terminals, TS-Anwendungen und Drucker, die immer mit der Anwendung verbunden sein sollen, können Sie den automatischen Verbindungsaufbau beim Anwendungsstart veranlassen.

KDCLTERM, KDCPTERM

KC_MODIFY_OBJECT mit *obj_type*=KC_PTERM oder KC_LTERM

Informationen über die Verfügbarkeit von Clients und Druckern lesen

Mit den Informationsfunktionen von openUTM können Sie Informationen über die Verfügbarkeit von Clients und Druckern abfragen. Es werden folgende Informationen zur Verfügung gestellt:

- Derzeitiger Status des Client/Druckers (ist er z. Z. gesperrt oder nicht)
- Besteht derzeit eine Verbindung bzw. wird gerade versucht eine Verbindung aufzubauen
- Zeit, die der Drucker bzw. der Client bereits mit der Anwendung verbunden ist
- Anzahl der auf der Verbindung ausgetauschten Nachrichten
- Anzahl der Ausfälle der Verbindung zum Client/Drucker
- Schwellwert der Message Queue (*qlev*)
- Anzahl der Aufträge in der Message Queue eines Druckers/Druckerbündels, bei der eine Verbindung zum Drucker(bündel) automatisch aufgebaut wird.

KDCINF LTERM oder PTERM

KC_GET_OBJECT mit *obj_type*=KC_PTERM oder KC_LTERM

4 Konfiguration dynamisch ändern

openUTM stellt Ihnen an der Programmschnittstelle zur Administration Funktionen zur Verfügung, mit denen Sie während des Anwendungslaufs neue Objekte in die Konfiguration der Anwendung eintragen bzw. aus der Konfiguration löschen können.

Durch diese Funktionen wird die Verfügbarkeit von UTM-Anwendungen weiter erhöht. Eine Neugenerierung der Anwendung mit KDCDEF, für die der Betrieb unterbrochen werden muss, ist weitaus seltener erforderlich. Darüber hinaus wird die Neugenerierung einer UTM-Anwendung erleichtert und ihr Aufwand minimiert. Entsprechende Empfehlungen für die Neugenerierung einer UTM-Anwendung finden Sie im Abschnitt „Empfehlungen für die Neugenerierung einer Anwendung".

Mit den UTM-Funktionen zum dynamischen Ändern der Konfiguration können Sie folgende Objekte eintragen und löschen:

- Benutzerkennungen einschließlich zugehöriger Queues,
- Keysets,
- Transportverbindungen zu entfernten LU6.1-Anwendungen,
- LU6.1-Sessions,
- Transaktionscodes der eigenen Anwendung,
- Transaktionscodes, über die Service-Programme in Partner-Anwendungen gestartet werden,
- LTERM-Partner,
- Clients, Drucker,
- Teilprogramme und VORGANG-Exits,
  (nur in Anwendungen mit Lademodulen, Shared Objects, bzw. DLLs)
- TAC-Queues.

Objekte in die Konfiguration eintragen und Objekte löschen können Sie entweder mit Hilfe der Administrations-Tools WinAdmin und WebAdmin oder mit Hilfe von Administrationsprogrammen, die Sie selbst erstellen. Zum Eintragen neuer Objekte steht Ihnen an der Programmschnittstelle zur Administration der Aufruf KC_CREATE_OBJECT zur Verfügung. Mit dem Aufruf KC_DELETE_OBJECT können Sie Objekte aus der Konfiguration löschen. Mit dem Aufruf KC_MODIFY_OBJECT haben Sie die Möglichkeit, einzelne Objekteigenschaften zu verändern.

> **i** Die Funktionen zum dynamischen Ändern der Konfiguration sind auch in der Funktionsvariante UTM-F in vollem Umfang nutzbar. openUTM sichert die Änderungen der Konfiguration, die sich auf das Eintragen, Löschen und Modifizieren dynamischer Objekte beziehen, in der KDCFILE. Die geänderten Konfigurationsdaten sind dann für den nächsten Anwendungslauf verfügbar.

Im Folgenden ist beschrieben, was Sie bei der KDCDEF-Generierung der Anwendung beachten müssen, wenn Sie im laufenden Betrieb Objekte in die Konfiguration eintragen wollen, und was Sie beim dynamischen Eintragen von Objekten sowie beim Löschen von Objekten aus der Konfiguration der Anwendung beachten müssen.

4.1 Anforderungen an die KDCDEF-Generierung

Damit Sie Objekte dynamisch in die Konfiguration Ihrer UTM-Anwendung aufnehmen können, müssen Sie bei der Generierung der Anwendung mit KDCDEF die im Folgenden beschriebenen Vorbereitungen treffen.
Für das Löschen von Objekten aus der Konfiguration sind keine Vorbereitungen bei der KDCDEF-Generierung nötig.

Plätze in den Objekttabellen der KDCFILE reservieren

Die Konfigurationsdaten einer UTM-Anwendung werden in den Objekttabellen der KDCFILE abgelegt, die bei der KDCDEF-Generierung der Anwendung erzeugt wird. Bei der KDCDEF-Generierung wird auch der für diese Tabellen zur Verfügung stehende Platz festgelegt. Deshalb müssen Sie für Objekte, die Sie erst im laufenden Betrieb in die Konfiguration der Anwendung aufnehmen wollen, bei der KDCDEF-Generierung leere Tabellenplätze reservieren. Dazu dient die KDCDEF-Anweisung RESERVE (siehe openUTM-Handbuch „Anwendungen generieren “).

In der RESERVE-Anweisung geben Sie an, wieviele leere Tabellenplätze für jeden einzelnen Objekttyp angelegt werden sollen, d.h. Sie geben an, wieviele LTERM-Partner dynamisch eintragbar sein sollen, wieviele Transaktionscodes usw. Die Reservierung der Tabellenplätze erfolgt Objekttyp-spezifisch, d.h. ein Tabellenplatz, den Sie z.B. für einen LTERM-Partner reserviert haben, kann nicht von einem Transaktionscode belegt werden usw.

Während des Anwendungslaufs können Sie genau so viele Objekte eines Typs dynamisch eintragen, wie leere Tabellenplätze mit KDCDEF reserviert wurden. Durch das Löschen eines anderen Objekts vom gleichen Typ wird i. a. kein Tabellenplatz für ein neues Objekt frei. Eine Ausnahme bilden Benutzerkennungen und Verbindungen für die verteilte Verarbeitung über LU6.1 bei stand-alone-Anwendungen. Diese können Sie durch „sofortiges Löschen“ aus der Konfiguration herausnehmen (siehe Abschnitt „Objekte dynamisch aus der Konfiguration löschen"). Die Tabellenplätze dieser Benutzerkennungen bzw. LU6.1-Verbindungen werden dann sofort freigegeben und können durch neue Benutzerkennungen bzw. LU6.1-Verbindungen belegt werden.

Bei der Reservierung der Tabellenplätze mit RESERVE ist Folgendes zu beachten:

Für jeden UPIC-Client und für jede TS-Anwendung (Client vom Typ APPLI oder SOCKET), die Sie dynamisch in die Konfiguration eintragen, erzeugt openUTM intern eine Benutzerkennung. In UTM-Anwendungen, die mit Benutzerkennungen generiert sind (d.h. die KDCDEF-Generierung enthält mindestens eine USER-Anweisung), wird deshalb für jeden dynamisch eingetragenen Client vom Typ APPLI, SOCKET oder UPIC zusätzlich ein reservierter Tabellenplatz für Benutzerkennungen belegt. Diese Tabellenplätze werden beim Löschen der Clients nicht freigegeben. In Anwendungen ohne Benutzerkennungen werden diese Tabellenplätze von openUTM intern reserviert.

Zum Reservieren der Tabellenplätze siehe openUTM-Handbuch „Anwendungen generieren“, Steueranweisung RESERVE.

Lockcodes, BCAMAPPL-Namen und Formatierungssystem generieren

Im KDCDEF-Lauf müssen Sie bereits bestimmte Objekte oder Werte statisch vorgenerieren, wenn Sie diese später bei der dynamischen Konfiguration referenzieren wollen; Beispiele hierfür sind der Wertebereich von Lockcodes und die Namen der Transportsystemzugangspunkte der lokalen Anwendung.

- Lockcodes (Zugriffsschutz), die Sie den Transaktionscodes und LTERM-Partnern zuordnen wollen, müssen in dem Bereich zwischen 1 und dem in KEYVALUE (MAX-Anweisung) festgelegten Maximalwert liegen. Deswegen sollten Sie den Wert von KEYVALUE hinreichend groß wählen. Das Lock-/Keycode-Konzept ist ausführlich im openUTM-Handbuch „Konzepte und Funktionen“ beschrieben.
- Alle Namen der lokalen Anwendung (BCAMAPPL-Namen), über die die Verbindungen zu Clients oder Druckern aufgebaut werden sollen, müssen mit KDCDEF generiert werden. Denken Sie insbesondere daran, dass für die Anbindung von TS-Anwendungen über die Socket-Schnittstelle oder HTTP-Clients (PTYPE=SOCKET) eigene BCAMAPPL-Namen generiert werden müssen.
- Nur auf BS2000-Systemen: Sollen Benutzerkennungen und LTERM-Partnern Startformate zugeordnet werden, dann muss bei der KDCDEF-Generierung ein Formatierungssystem generiert werden (Anweisung FORMSYS). Werden #Formate als Startformate verwendet, dann muss zusätzlich ein Anmelde-Vorgang generiert werden.

Voraussetzungen für das Eintragen von Teilprogrammen und VORGANG-Exits

Neue Teilprogramme und VORGANG-Exits können Sie nur dann dynamisch in die Konfiguration der Anwendung aufnehmen, wenn die Anwendung folgende Voraussetzung erfüllt:

- Eine UTM-Anwendung auf BS2000-Systemen muss mit Lademodulen generiert werden (KDCDEF-Generierung mit LOAD-MODULE-Anweisungen). Das Teilprogramm darf jedoch nicht in ein Lademodul gebunden werden, das statisch ins Anwendungsprogramm eingebunden ist (Lademodus STATIC).
- Eine UTM-Anwendung auf einem Unix- oder Linux-System muss mit Shared Objects generiert werden (KDCDEF-Generierung mit SHARED-OBJECT-Anweisungen).
- Eine UTM-Anwendung auf einem Windows-System muss Windows-DLLs verwenden. Mehr Informationen zur Generierung finden Sie im openUTM-Handbuch „Anwendungen generieren“.

Ein Teilprogramm, das Sie im laufenden Betrieb neu eintragen wollen, muss in ein Lademodul, Shared Object bzw. eine DLL gebunden werden, das/die bei der KDCDEF-Generierung definiert wurde.

Für jede Programmiersprache, in der Sie Teilprogramme Ihrer Anwendung erstellen wollen, muss mindestens ein Teilprogramm mit KDCDEF generiert werden. Nur dann sind die für den Ablauf benötigten Sprachanschlussmodule und Laufzeitsysteme im Anwendungsprogramm enthalten.

Hinweis für BS2000-Systeme:

Für Teilprogramme, die mit ILCS-fähigen Compilern (COMP=ILCS) übersetzt werden, genügt es, wenn Sie bei der KDCDEF-Generierung ein Teilprogramm mit COMP=ILCS generieren. Es müssen keine PROGRAM-Anweisungen für die verschiedenen Programmiersprachen abgesetzt werden.

> **i** Für COBOL Programme muss der betreffende LOAD-MODULE mit ALTERNATE-LIBRARIES=YES generiert werden, damit die benötigten RTS-Module per Autolink nachgeladen werden können.

Voraussetzungen für das dynamische Eintragen von Transaktionscodes

Wenn Sie Transaktionscodes dynamisch in die Konfiguration eintragen wollen, müssen Sie Folgendes beachten:

- Transaktionscodes für Teilprogramme, die eine X/Open-Programmschnittstelle nutzen, können nur dynamisch erzeugt werden, wenn bei der KDCDEF-Generierung mindestens ein Transaktionscode für ein X/Open-Teilprogramm generiert wurde (TAC-Anweisung mit API!=KDCS).
- Wollen Sie die Transaktionscodes in TAC-Klassen einteilen, um die Auftragsbearbeitung steuern zu können, dann müssen Sie bei der KDCDEF-Generierung mindestens eine TAC-Klasse erzeugen.

  TAC-Klassen können Sie bei der KDCDEF-Generierung auf drei Arten erzeugen:

  1. Sie generieren einen Transaktionscode, für den Sie im Operanden TACCLASS (TAC-Anweisung) eine TAC-Klasse angeben. Die angegebene TAC-Klasse wird dann von KDCDEF implizit erzeugt.
  2. Wenn Sie die Anwendung ohne Prioritätensteuerung betreiben (die Anwendung enthält keine TAC-PRIORITIES-Anweisung) dann können Sie TAC-Klassen erzeugen, indem Sie eine TACCLASS-Anweisung schreiben.
  3. Sie können implizit TAC-Klassen erzeugen, indem Sie eine TAC-PRORITIES-Anweisung schreiben.

  Haben Sie bei der KDCDEF-Generierung eine TAC-Klasse erzeugt, dann können Sie die Transaktionscodes, die Sie dynamisch eintragen, jeder beliebigen TAC-Klasse zwischen 1 und 8 (Dialog) bzw. 9 und 16 (Asynchron) zuordnen. Die TAC-Klassen werden von openUTM implizit erzeugt; sie sind administrierbar.

  Ist die Anwendung ohne TAC-PRIORITIES generiert, dann legt openUTM bei implizit erzeugten TAC-Klassen die Prozessanzahl (TASKS) wie folgt fest:
  1 für Dialog-TAC-Klassen (Klasse 1 bis 8),
  und 0 für Asynchron-TAC-Klassen (Klasse 9 bis 16).

  Asynchron-TAC-Klassen werden von openUTM jedoch nur erzeugt, wenn Sie bei der KDCDEF-Generierung in der MAX-Anweisung ASYNTASKS > 0 gesetzt haben.

  In Anwendungen mit TAC-Klassen ohne Prioritätensteuerung können Sie Transaktionscodes, die Teilprogrammläufe mit blockierenden Aufrufen starten, nur dann dynamisch erzeugen, wenn TAC-Klassen mit PGWT=YES (Dialog- und/oder Asynchron-TAC-Klasse) bei der KDCDEF-Generierung explizit mit TACCLASS-Anweisungen generiert wurden. Zusätzlich muss MAX TASKS-IN-PGWT > 0 gesetzt werden.
- In Anwendungen mit Prioritätensteuerung (mit TAC-PRIORITIES-Anweisung) können Sie Transaktionscodes, die Teilprogrammläufe mit blockierenden Aufrufen starten (*kc_tac_str.pgwt*='Y'), nur dann dynamisch erzeugen, wenn bei der KDCDEF-Generierung MAX TASKS-IN-PGWT>0 gesetzt wurde.

Voraussetzungen für das dynamische Eintragen von Benutzerkennungen

Benutzerkennungen können Sie nur dynamisch in die Konfiguration aufnehmen, wenn Ihre Anwendung mit Benutzerkennungen generiert wird. Dazu muss Ihre KDCDEF-Generierung mindestens eine USER-Anweisung enthalten und mindestens eine Benutzerkennung muss Administrationsberechtigung haben (USER mit PERMIT=ADMIN).

Hinweis für BS2000-Systeme:

Sollen im Betrieb auch neue Benutzerkennungen mit Ausweiskarte in die Konfiguration eingetragen werden, dann müssen Sie dies beim Reservieren explizit angeben. Dazu geben Sie in der RESERVE-Anweisung an, wieviel Prozent der Tabellenplätze, die für Benutzerkennungen angelegt werden, von Benutzerkennungen mit Ausweiskarte belegt werden können (Operand CARDS in der RESERVE-Anweisung).

Sollen im Betrieb auch Benutzerkennungen mit Kerberos-Authentisierung dynamisch erzeugt werden, müssen diese mit dem Operanden PRINCIPALS der RESERVE-Anweisung reserviert werden.

4.2 Objekte dynamisch in die Konfiguration eintragen

Mit dem Aufruf KC_CREATE_OBJECT können Sie während eines Anwendungslaufs neue Objekte in die Konfiguration Ihrer Anwendung eintragen.

KC_CREATE_OBJECT, siehe "KC_CREATE_OBJECT - Objekte in die Konfiguration eintragen"

Pro KC_CREATE_OBJECT-Aufruf können Sie genau ein Objekt eintragen. Sie können jedoch innerhalb eines Administrationsprogramms KC_CREATE_OBJECT mehrfach aufrufen, um mehrere Objekte einzutragen. Beim Aufruf geben Sie den Typ des Objektes, seinen Namen und die Eigenschaften an, die das Objekt haben soll.

Das Eintragen der Objekte unterliegt der Transaktionssicherung. Erst nach erfolgreicher Transaktionssicherung werden die Konfigurationsdaten eines Objektes in die Objekttabelle geschrieben. Auf das in einem Teilprogramm erzeugte Objekt kann somit erst nach erfolgreichem Abschluss der Transaktion zugegriffen werden. Vorher kann das Objekt weder verwendet werden noch können die Eigenschaften des Objektes gelesen oder verändert werden. Aufrufe wie KC_MODIFY_OBJECT oder KC_GET_OBJECT können also erst nach erfolgreichem Abschluss des Neueintragens, d.h. nach erfolgreichem Abschluss der Transaktion, für das neue Objekt abgesetzt werden.

Innerhalb der Transaktion, in der ein Objekt erzeugt wird, kann lediglich auf dieses Objekt zugegriffen werden, um eine Beziehung zu einem anderen Objekt herzustellen, das in derselben Transaktion erzeugt wird. Eine Beziehung wird z.B. hergestellt zwischen einem Client bzw. Drucker und seinem Anschlusspunkt, dem LTERM-Partner, zwischen einem Transaktionscode und dem zugehörigen Teilprogramm, zwischen einem Transaktionscode und seinem VORGANG-Exit und zwischen einem Keyset und den Objekten (wie LTERMs, USERs, TACs oder LTACs), die dieses Keyset referenzieren.

Werden zwei Objekte, die sich aufeinander beziehen, innerhalb einer Transaktion erzeugt, dann müssen Sie auf die Reihenfolge achten, in der die Objekte erzeugt werden. Zum Beispiel können Sie einen Client zusammen mit seinem Anschlusspunkt, dem LTERM-Partner, in einer Transaktion eintragen. Der LTERM-Partner muss jedoch vor dem Client erzeugt werden, da beim Eintragen des Client der Name des LTERM-Partners angegeben wird.

Allgemein müssen alle Objekte, auf die Sie sich beim Eintragen eines neuen Objektes beziehen, entweder bereits in der Konfiguration enthalten sein oder in derselben Transaktion vor dem neuen Objekt erzeugt werden. Im Folgenden wird für die einzelnen Objekttypen detailliert beschrieben, in welcher Reihenfolge die Objekte eingetragen werden müssen.

UTM-Cluster-Anwendungen (Unix-, Linux- und Windows-Systeme)

In UTM-Cluster-Anwendungen gilt:
Der Aufruf wirkt Cluster-global, d.h. in allen Knoten-Anwendungen werden die Objekte dynamisch in die Konfiguration eingetragen.

Verfügbarkeit dynamisch eingetragener Objekte

Dynamisch eingetragene Objekte sind auch in folgenden Anwendungsläufen Bestandteil der Konfiguration, sofern Sie nicht mit KC_DELETE_OBJECT gelöscht werden. Das gilt sowohl für Objekte einer UTM-S- als auch einer UTM-F-Anwendung.

4.2.1 Clients, Drucker und LTERM-Partner eintragen

Zum Eintragen eines Client oder Druckers müssen Sie KC_CREATE_OBJECT mit Objekttyp KC_PTERM aufrufen. Zum Eintragen eines LTERM-Partners müssen Sie den Objekttyp KC_LTERM angeben.

> ¡ In UTM-Anwendungen auf Windows-Systemen werden Drucker nicht unterstützt.

Damit sich ein Client oder Drucker an die Anwendung anschließen kann, muss ihm ein LTERM-Partner zugeordnet werden. Geben Sie diesen LTERM-Partner beim Eintragen des Client bzw. Druckers an, dann muss der LTERM-Partner entweder bereits in der Konfiguration der Anwendung existieren oder in derselben Transaktion vor dem Client/Drucker eingetragen werden. Es gilt also allgemein die Regel:

LTERM-Partner (KC_LTERM) vor Client/Drucker (KC_PTERM)

Beim Eintragen von Druckern/Clients sind folgende Fälle zu unterscheiden:

- Terminals und Druckern,
- TS-Anwendungen und UPIC-Clients.

Terminals und Drucker

Terminals und Drucker können Sie in die Konfiguration eintragen, ohne ihnen direkt einen LTERM-Partner zuzuordnen. In dem Fall müssen Sie also beim Eintragen keinen LTERM-Partner angeben. Den LTERM-Partner können Sie dem Terminal/Drucker dann zu einem späteren Zeitpunkt zuordnen. Dazu stehen Ihnen das Administrationskommando KDCSWTCH und der Aufruf KC_MODIFY_OBJECT (Objekttyp KC_PTERM) zur Verfügung. Die Zuordnung muss dann in einer eigenen Transaktion erfolgen.

Geben Sie beim Eintragen eines Terminals oder Druckers jedoch einen LTERM-Partner an, dann muss dieser nach obiger Regel bereits in der Konfiguration der Anwendung existieren oder in derselben Transaktion wie das Terminal bzw. der Drucker vor dem Terminal/Drucker eingetragen werden.

Einem Drucker können Sie einen LTERM-Partner zuordnen, der bereits einem anderen Drucker zugeordnet ist. Die alte Zuordnung wird nicht aufgehoben. Ein LTERM-Partner kann mehreren Druckern zugeordnet sein. Die Drucker bilden dann ein Druckerbündel und arbeiten die Message Queue des LTERM-Partners gemeinsam ab.

Einem Terminal können Sie nur einen LTERM-Partner zuordnen, der keinem anderen Client zugeordnet ist. Eine eventuell existierende Zuordnung zu einem anderen Terminal muss vor dem Erzeugen des Client in einer eigenen Transaktion aufgehoben werden (mit dem Administrationskommando KDCSWTCH oder dem Aufruf KC_MODIFY_OBJECT).

Soll für den Anschluss eines Terminals ein LTERM-Partner mit automatischem KDCSIGN erzeugt werden, dann müssen Sie dem LTERM-Partner die Benutzerkennung beim Eintragen zuordnen, für die das automatische KDCSIGN beim Verbindungsaufbau durchgeführt werden soll. Die Benutzerkennung muss vor dem Eintragen des LTERM-Partners in der Konfiguration enthalten sein oder in derselben Transaktion vor dem LTERM-Partner eingetragen werden. Allgemein gilt die Regel:

Benutzerkennung (KC_USER) vor LTERM-Partner (KC_LTERM)
vor Terminal (KC_PTERM)

Auf BS2000-Systemen gilt allgemein: Die Eigenschaft *usage_type* (D für Dialog-Partner oder O für Ausgabemedium) des LTERM-Partners muss mit dem Wert übereinstimmen, den Sie beim Eintragen des Client /Druckers in *usage* angeben.

Wird ein LTERM-Partner für einen Drucker erzeugt, der über ein Druckersteuer-LTERM (CTERM) administriert werden soll, dann müssen Sie dem LTERM-Partner das Druckersteuer-LTERM beim Eintragen zuordnen. Das Druckersteuer-LTERM muss vor dem Eintragen des LTERM-Partners entweder bereits in der Konfiguration der Anwendung enthalten sein (statisch oder dynamisch eingetragen) oder in derselben Transaktion wie der LTERM-Partner, jedoch vor dem LTERM-Partner, eingetragen werden. Es gilt die Regel:

Druckersteuer-LTERM (KC_LTERM) vor LTERM-Partner (KC_LTERM) vor Drucker (KC_PTERM)

TS-Anwendungen und UPIC-Clients

TS-Anwendungen oder UPIC-Clients (Clients vom Typ APPLI, SOCKET, UPIC-R und UPIC-L) müssen Sie bereits beim Eintragen einen LTERM-Partner zuordnen. Dieser LTERM-Partner muss in derselben Transaktion wie der Client erzeugt werden, jedoch vor dem Client. D.h. der KC_CREATE_OBJECT-Aufruf, der den LTERM-Partner erzeugt, muss vor dem KC_CREATE_OBJECT-Aufruf, der den Client einträgt, in derselben Transaktion bearbeitet werden. Es gilt in diesem Fall die Regel:

LTERM-Partner (KC_LTERM) vor der TS-Anwendung/UPIC-Client (KC_PTERM) in derselben Transaktion

Die Zuordnung Client zu LTERM-Partner kann nicht aufgehoben werden, solange der Client in der Konfiguration enthalten ist.

Für einen LTERM-Partner eines derartigen Clients benötigt openUTM eine fest zugeordnete Benutzerkennung, die Verbindungs-Benutzerkennung.

Die Verbindungs-Benutzerkennung können Sie explizit generieren. In diesem Fall muss sie in derselben Transaktion wie LTERM-Partner und Client eingetragen werden. Die Benutzerkennung muss jedoch *vor* dem Client in die Konfiguration eingetragen werden. Bei der Zuordnung der Benutzerkennung zum LTERM-Partner sind folgende Fälle zu unterscheiden:

- Sie erzeugen explizit eine Benutzerkennung mit dem Namen des LTERM-Partners. Dann erfolgt die Zuordnung automatisch beim Eintragen des LTERM-Partners.
- Sie erzeugen eine Benutzerkennung mit einem beliebigen Namen. Dann müssen Sie den Namen beim Eintragen des LTERM-Partners explizit angeben (Feld *kc_lterm_str.user_gen*).

Wenn Sie die Verbindungs-Benutzerkennung nicht explizit generieren, dann erzeugt openUTM implizit eine Benutzerkennung mit dem Namen des LTERM-Partners.

Die Verbindungs-Benutzerkennung ist immer für diesen Client reserviert. Es kann sich kein anderer Benutzer oder Client mit dieser Benutzerkennung bei der Anwendung anmelden.

Für diese Benutzerkennung wird einer der reservierten Tabellenplätze belegt. Ist kein Tabellenplatz für eine Benutzerkennung mehr frei, dann werden LTERM-Partner und Client nicht in die Konfiguration eingetragen. Die KC_CREATE_OBJECT-Aufrufe werden abgewiesen.

Allgemein gilt:
In Anwendungen mit Benutzerkennungen benötigen Sie zum Eintragen eines Clients vom Typ APPLI, SOCKET oder UPIC-R/UPIC-L drei reservierte Tabellenplätze: einen für den Objekttyp PTERM, einen für den Objekttyp LTERM und einen für den Objekttyp USER.

Beim Eintragen gilt folgende Regel:

> Benutzerkennung (KC_USER) vor LTERM-Partner (KC_LTERM) vor TS-Anwendung/UPIC-Client (KC_PTERM)
> Die drei Objekte müssen innerhalb derselben Transaktion eingetragen werden

Eine Verbindungs-Benutzerkennung ist nicht administrierbar. Nach dem Eintragen der Benutzerkennung können ihre Eigenschaften nicht mehr geändert werden.

Beispiel für das Eintragen einer TS-Anwendung bzw. eines UPIC-Clients

Ein Programm, das eine TS-Anwendung bzw. einen UPIC-Client einträgt und ihr/ihm explizit eine Verbindungs-Benutzerkennung zuordnet, muss den im folgenden Schema dargestellten Aufbau haben. Die in eckige Klammern gesetzten KDCS-Aufrufe sind optional. Insbesondere können die einzelnen KC_CREATE_OBJECT-Aufrufe in verschiedenen KDCS-Programmen liegen. Die Programme müssen jedoch in derselben Transaktion ablaufen (Programmabschluss z.B. mit PEND PA).

```
#include <kcadminc.h>                 /* Definitionen aufnehmen           */

INIT                                  /* KDCS-Aufruf zum Anmelden bei UTM */

[MGET]                                /* KDCS-Aufruf zum Einlesen des     */
                                      /* aufrufenden TACs und der         */
                                      /* übergebenen Parameter            */

KC_CREATE_OBJECT mit obj_type=KC_USER  /* KDCADMI-Aufruf zum Eintragen der */
                                      /* Benutzerkennung                  */

/* eventuell Fehlerbehandlung: Der folgende KC_CREATE_OBJECT-Aufruf sollte */
/* nur abgesetzt werden, wenn der vorherige Aufruf fehlerfrei war.         */

KC_CREATE_OBJECT mit obj_type=KC_LTERM /* KDCADMI-Aufruf zum Eintragen des */
                                      /* LTERM-Partners.                  */
/* eventuell Fehlerbehandlung                                           */

KC_CREATE_OBJECT mit obj_type=KC_PTERM /* KDCADMI-Aufruf zum Eintragen des */
                                      /* Client                           */
/* eventuell Fehlerbehandlung                                           */
MPUT                                  /* KDCS-Aufruf zum Senden einer     */
....                                  /* Nachricht an den Auftraggeber    */
PEND FI / RE / SP / FC                /* KDCS-Aufruf zum Beenden der      */
                                      /* Transaktion                      */
```

4.2.2 Teilprogramme, Transaktionscodes, TAC-Queues und VORGANG-Exits eintragen

Zum Eintragen eines neuen Teilprogramms oder VORGANG-Exits müssen Sie KC_CREATE_OBJECT für den Objekttyp KC_PROGRAM aufrufen.
Zum Eintragen eines neuen Transaktionscodes oder einer neuen TAC-Queue müssen Sie den Objekttyp KC_TAC angeben.

Neue Teilprogramme und VORGANG-Exits können Sie nur dynamisch eintragen, wenn die Anwendung mit Lademodulen (BS2000-Systeme), Shared Objects (Unix- und Linux-Systeme), bzw. DLLs (Windows-Systeme) generiert wurde.

Einem Teilprogramm sollten Sie mindestens einen Transaktionscode zuordnen, damit es aufgerufen werden kann. Den Transaktionscode dürfen Sie erst nach dem Teilprogramm in die Konfiguration aufnehmen. Das Teilprogramm muss also entweder zum Zeitpunkt, zu dem der Transaktionscode mit KC_CREATE_OBJECT erzeugt wird, bereits in der Konfiguration der Anwendung enthalten sein oder in derselben Transaktion vor dem Transaktionscode eingetragen werden. Das Teilprogramm kann sowohl mit KDCDEF generiert als auch in einer eigenen Transaktion eingetragen worden sein.

Sie können also auch bereits in der Konfiguration enthaltenen Teilprogrammen neue Transaktionscodes zuordnen.

Ein neu erzeugtes Teilprogramm kann erst aufgerufen werden, wenn es geladen und ihm mindestens ein Transaktionscode zugeordnet wurde. Zum Laden muss das Teilprogramm übersetzt und in ein mit KDCDEF generiertes Lademodul, Shared Object bzw. DLL der Anwendung eingebunden werden. Anschließend muss dieses Lademodul, Shared Object oder die DLL ausgetauscht werden (siehe KDCPROG im Abschnitt "KDCPROG - Lademodule/Shared Objects/DLLs austauschen" oder KC_MODIFY_OBJECT mit *obj_type*=KC_LOAD_MODULE im Abschnitt "obj_type=KC_LOAD_MODULE").

Hinweis für BS2000-Systeme:

- Handelt es sich um eine Lademodul mit Slices und liegt der Public Slice des Lademoduls in einem Common Memory Pool, müssen Sie anschließend noch ein KDCAPPL PROG=NEW oder KC_CHANGE_APPLICATION absetzen, damit dieses Lademodul ausgetauscht wird. Erst dann können Sie den neuen oder geänderten Service nutzen.
- Ein neues Teilprogramm darf nicht in ein Lademodul eingebunden werden, das statisch zum Anwendungsprogramm gebunden ist (Lademodus STATIC).

Soll einem Transaktionscode, den Sie dynamisch eintragen, ein VORGANG-Exit zugeordnet werden (*kc_tac_str. exit_name*), dann muss dieser VORGANG-Exit bereits in der Konfiguration der Anwendung existieren oder er muss in derselben Transaktion wie der Transaktionscode eingetragen werden, jedoch vor dem Transaktionscode.

Damit der VORGANG-Exit durchlaufen werden kann, muss das zugehörige Programm geladen sein. Neu eingetragene VORGANG-Exits müssen also wie Teilprogramme in ein Lademodul, Shared Object oder eine DLL der Anwendung eingebunden werden, das dann ausgetauscht werden muss.

Für das Eintragen von Teilprogrammen, Transaktionscodes und VORGANG-Exits gilt allgemein die Regel:

Teilprogramm(KC_PROGRAM) und VORGANG-Exit (KC_PROGRAM)
vor Transaktionscodes (KC_TAC)

> Die Transaktionscodes der Event-Services BADTAC, MSGTAC und SIGNON (KDCBADTC, KDCMSGTC, KDCSGNTC) können nicht dynamisch in die Konfiguration eingetragen werden.

4.2.3 Benutzerkennungen eintragen

Zum Eintragen einer neuen Benutzerkennung und einer zugehörigen USER-Queue müssen Sie KC_CREATE_OBJECT für den Objekttyp KC_USER aufrufen. Benutzerkennungen, die bestimmten LTERM-Partnern für ein automatisches KDCSIGN fest zugeordnet werden sollen, müssen vor dem Eintragen des LTERM-Partners erzeugt werden. Was beim Eintragen dieser Benutzerkennungen zu beachten ist, ist im Abschnitt „Clients, Drucker und LTERM-Partner eintragen“ beschrieben.

4.2.4 Keysets erzeugen

Zum Erzeugen eines neuen Keysets müssen Sie KC_CREATE_OBJECT für den Objekttyp KC_KSET aufrufen. Das neue Keyset können Sie anschließend in der gleichen Transaktion einer neuen Benutzerkennung, einem neuen LTERM-Partner, einem neuen Transaktionscode bzw. einer TAC-Queue oder einem neuen LTAC zuordnen.

Es gilt die Regel:

> Keyset (KC_KSET) vor LTERM-Partner (KC-LTERM)
> und Benutzerkennung (KC-USER) und Transaktionscode (KC_TAC)
> und LTAC (KC_LTAC)

4.2.5 LU6.1 - Verbindungen für verteilte Verarbeitung eintragen

Bei einer Kopplung über das LU6.1-Protokoll müssen Sie für die Kommunikation zwischen der lokalen UTM-Anwendung und einer fernen Anwendung eine oder mehrere Transportverbindungen und Sessions definieren, über die die Kommunikationsbeziehungen hergestellt werden.

Für den Eintrag einer Transportverbindung rufen Sie KC_CREATE_OBJECT für den Objekttyp KC_CON auf. Um eine Session zu definieren, rufen Sie KC_CREATE_OBJECT für den Objekttyp KC_LSES auf.

Voraussetzung ist, dass in jeder Anwendung bereits LPAP-Partner bekannt und Sessioneigenschaften definiert sind.

Zu jedem LPAP muss eine Anzahl von CON- und LSES-Objekten erzeugt werden; die Anzahl der CON- bzw. LSES-Objekte bestimmt die Anzahl der parallelen Verbindungen, die über einen LPAP mit einer Partner-Anwendung möglich sind.

In Cluster-Anwendungen (Unix-, Linux- und Windows-Systeme) müssen zu jedem CON-Objekt so viele LSES-Objekte erzeugt werden wie es Knoten-Anwendungen gibt, damit eine Partner-Anwendung mit allen Knoten-Anwendungen kommunizieren kann.

Für jede parallele Verbindung über einen LPAP wird ein CON- und ein LSES-Objekt erzeugt und dem LPAP zugeordnet. Dabei muss jedes CON- und jedes LSES-Objekt in jeder der beteiligten Anwendungen korrespondierend erzeugt werden. Das bedeutet:

- ein CON-Name in der lokalen Anwendung ist gleich einem BCAMAPPL-Namen in der fernen Anwendung und umgekehrt,
- ein LSES-Name in der lokalen Anwendung ist gleich einem RSES-Namen in der fernen Anwendung und umgekehrt.

> **VORSICHT!**
> Es ist nicht erlaubt, für einen LPAP-Namen mehrere CON-Objekte zu erzeugen, die zu unterschiedlichen Anwendungen führen oder die über ihre korrespondierenden CON-Objekte in der Partner-Anwendung unterschiedlichen LPAPs zugeordnet sind.
>
> Solche Konfigurationen werden von openUTM nicht erkannt und führen beim Verbindungs- bzw. Sessionaufbau und beim Sessionwiederanlauf zu Fehlern.

4.2.6 LTACs eintragen

Um in der lokalen Anwendung dynamisch einen Transaktionscode zu erzeugen, mit dem ein Vorgang bzw. ein fernes Service-Programm in einer Partner-Anwendung gestartet wird, müssen Sie KC_CREATE_OBJECT für den Objekttyp KC_LTAC aufrufen.

Dem lokalen Transaktionscode wird entweder

- (bei einstufiger Adressierung) der Name eines Transaktionscodes in einer bestimmten Partner-Anwendung zugeordnet. Dadurch adressiert der lokale Transaktionscode sowohl die Partner-Anwendung als auch den Transaktionscode in dieser Anwendung, oder
- (bei zweistufiger Adressierung) der Name eines Transaktionscodes in irgendeiner Partner-Anwendung zugeordnet. In welcher Partner-Anwendung das mit dem lokalen Transaktionscode angesprochene Service-Programm ablaufen soll, muss explizit an der Programmschnittstelle angegeben werden.

Sollen Zugriffsrechte über eine Access Liste erteilt werden, dann muss das dazu verwendete Keyset existieren oder zuvor dynamisch erzeugt werden; das dynamische Erzeugen des Keysets und des referenzierten LTACs kann auch innerhalb einer Transaktion erfolgen. Sollen die Zugriffsrechte über einen Lockcode gesteuert werden, dann darf der Zahlenwert für den Lockcode nicht kleiner als 1 und nicht größer als der in der Anwendung erlaubte Maximalwert sein (KDCDEF-Anweisung MAX, Operand KEYVALUE).

Es gilt die Regel:

Keyset (KC_KSET) vor LTAC (KC_LTAC)

4.2.7 Format und Eindeutigkeit der Objektnamen

Jedem Objekt, das Sie mit KC_CREATE_OBJECT dynamisch in die Konfiguration eintragen, müssen Sie einen Namen bzw. eine logische Adresse (Clients und Druckern) zuordnen. Über seinen Namen bzw. seine logische Adresse muss das Objekt innerhalb der Anwendung eindeutig identifizierbar sein. Folgende Regeln sind bei der Namensvergabe zu beachten.

- Sie dürfen keinen reservierten Namen benutzen. (--> Reservierte Namen)
- Der Name eines Objektes muss innerhalb der Namensklasse eindeutig sein, zu der das Objekt gehört. (--> Eindeutigkeit der Namen und Adressen)
- Die Namen dürfen die vorgeschriebene Maximallänge nicht überschreiten und nur bestimmte Zeichen enthalten (Format). (--> Format der Namen)

Die Namen von Objekten, die zuvor mit KC_DELETE_OBJECT verzögert gelöscht (zum Löschen vorgemerkt) wurden, dürfen für Objekte derselben Namensklasse nicht verwendet werden. Namen von Benutzerkennungen und Namen von Verbindungen für die verteilte Verarbeitung über LU6.1, die sofort gelöscht wurden, können sofort wieder vergeben werden.

Reservierte Namen

Namen von Transaktionscodes, die mit KDC beginnen, sind für die Transaktionscodes der Event-Services und der Admininistrationskommandos reserviert. Namen die mit KDC beginnen, dürfen also für andere Objekte **nicht** verwendet werden:

In UTM-Anwendungen auf BS2000-Systemen sollten Teilprogrammnamen nicht mit einem Präfix beginnen, das für Compiler-Laufzeitmodule verwendet wird (z.B. IT, IC).

In UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen sollten die Namen der Objekte nicht mit KC, x, ITS oder mF beginnen. Externe Namen (z.B. Teilprogrammnamen) sollten nicht mit 't_', 'a_', 'o_', 'p_' oder 's_' beginnen. 't_' ist für PCMX reserviert. 'a_', 'o_','p_' und 's_' sind für OSS reserviert.

Alle Namen, die auf einer bestimmten Plattformen reserviert sind, sollten auf den anderen Plattformen ebenfalls nicht verwendet werden, damit die Anwendung portierbar bleibt.

Eindeutigkeit der Namen und Adressen

Die Namen und Adressen der Objekte einer UTM-Anwendung sind in Namensklassen zusammengefasst. Innerhalb einer Namensklasse müssen die Objektnamen eindeutig sein, sie dürfen nicht mehreren Objekten zugeordnet sein. Es gibt drei Namensklassen:

Zur 1. Namensklasse gehören die Namen folgender Objekte:

- LTERM-Partner (Objekttyp KC_LTERM);
  dazu gehören auch die LTERM-Partner der LTERM-Pools.
- Transaktionscodes und TAC-Queues (Objekttyp KC_TAC).
- LPAP- bzw. OSI-LPAP-Partner für die Server-Server-Kommunikation (Objekttyp KC_LPAP und KC_OSI_LPAP).

Zur 2. Namensklasse gehören die Namen folgender Objekte:

- Benutzerkennungen einschließlich zugehöriger Queues (Objekttyp KC_USER)
- Sessions für die verteilte Verarbeitung über LU6.1 (Objekttyp KC_LSES)

- Verbindungen und Associations für die verteilte Verarbeitung über OSI TP (Objekttyp KC_OSI_ASSOCIATION)

Zur 3. Namensklasse gehören die Namen folgender Objekte:

- Clients und Drucker (Objekttyp KC_PTERM).
  Clients sind hierbei: Terminals, UPIC-Clients, TS-Anwendungen (DCAM-, CMX-Anwendungen und UTM-Anwendungen), die bei der Kommunikation das LU6.1- und das OSI TP-Protokoll nicht nutzen.
- Name der Partner-Anwendung bei der verteilten Verarbeitung über das Protokoll LU6.1 (Objekttyp KC_CON).
- Name der Partner-Anwendung bei der verteilten Verarbeitung über das Protokoll OSI TP.
  Auch wenn OSI-CONs nicht dynamisch erzeugt werden können, sind die bereits für OSI-CONs generierten Namen für diese Namensklasse vergeben und können nicht für andere Objekte dieser Namensklasse verwendet werden.
- Multiplexanschlüsse (Objekttyp KC_MUX, nur auf BS2000-Systemen).

Die in der 3. Namensklasse aufgeführten Objekte sind Kommunikationspartner der UTM-Anwendung. Sie bzw. die Verbindungen zu ihnen müssen für openUTM eindeutig identifizierbar sein. Deshalb wird jeder Kommunikationspartner über eine logische Adresse identifiziert. Die logische Adresse ist ein Namenstripel, das sich aus den folgenden Komponenten zusammensetzt:

1. Name des Kommunikationspartners (*pt_name*, *co_name* der LU6.1-Verbindung, *mx_name*). Das ist der symbolische Name, unter dem der Kommunikationspartner beim Transportsystem bekannt ist.
2. Name des Rechners, auf dem sich der Kommunikationspartner befindet (*pronam*).
3. Name der lokalen Anwendung, über den die Verbindung zum Kommunikationspartner aufgebaut wird (*bcamappl* oder ACCESS-POINT). Auch wenn OSI TP-Verbindungen nicht dynamisch erzeugt werden können, sind die bereits für ACESS-POINTs generierten Namen zu berücksichtigen.

Die Namenstripel der Kommunikationspartner müssen voneinander verschieden sein.

Format der Namen

Alle Namen, die Sie definieren, müssen folgenden Konventionen entsprechen:

- Die Namen von LTERM-Partnern, Clients und Druckern (KC_PTERM), Transaktionscodes, Benutzerkennungen, Keysets, LU6.1-Verbindungen und Sessions sowie Transaktionscodes für ferne Services dürfen 1 bis 8 Zeichen lang sein.
- Die Namen von Teilprogrammen dürfen, wenn die Anwendung mit Lademodulen/Shared Objects/DLLs generiert ist, bis zu 32 Zeichen lang sein.
- Erlaubte Zeichen für die Objektnamen in einer UTM-Anwendung auf BS2000-Systemen sind: A,B,C,...,Z, 0,1,..., 9, #, @, $. Es sind beliebige Kombinationen dieser Zeichen erlaubt.
- Erlaubte Zeichen für die Objektnamen in einer UTM-Anwendung auf Unix-, Linux- oder Windows-Systemen sind: A,B,C,...,Z, a,b,c,...,z, 0,1,...,9, #, @, $.

4.3 Objekte dynamisch aus der Konfiguration löschen

Mit dem Aufruf KC_DELETE_OBJECT der Programmschnittstelle zur Administration können Sie während eines Anwendungslaufs Objekte aus der Konfiguration Ihrer Anwendung löschen.

KC_DELETE_OBJECT, siehe "KC_DELETE_OBJECT - Objekte löschen"

Beim Löschen von Objekten sind zwei Arten zu unterscheiden: das verzögerte Löschen (delay) und das sofortige Löschen (immediate).

- *verzögertes Löschen* (KC_DELETE_OBJECT *subopcode1*=KC_DELAY)

  Verzögertes Löschen heißt, dass die Objekte als gelöscht gekennzeichnet werden. Die Objekte und deren Eigenschaften bleiben in der Objekttabelle stehen. Das verzögerte Löschen wirkt wie eine dauerhafte Sperre, die nicht zurückgenommen werden kann. Das eigentliche Löschen der Objekte aus den Objekttabellen erfolgt erst bei der Neugenerierung, wenn mit dem inversen KDCDEF gearbeitet wird.

  Auf ein verzögert gelöschtes Objekt kann kein Benutzer mehr zugreifen. Lediglich die Administration kann noch lesend auf verzögert gelöschte Objekte zugreifen. D.h. es können mit KC_GET_OBJECT oder mit dem Administrationskommando KDCINF Namen und Eigenschaften „verzögert gelöschter“ Objekte gelesen werden. Die Eigenschaften gelöschter Objekte können aber nicht mehr verändert werden. Eine „verzögert gelöschte“ Benutzerkennung kann jedoch durch „sofortiges Löschen“ vollständig aus der Konfiguration entfernt werden.

  Durch das verzögerte Löschen eines Objektes wird kein Platz in der Objekttabelle freigegeben. Die Namen dieser Objekte bleiben belegt, d.h. es können innerhalb ihrer Namensklasse keine neuen Objekte mit gleichem Namen dynamisch eingetragen werden. Insbesondere können keine neuen Objekte mit gleichem Namen und gleichem Objekttyp dynamisch erzeugt werden.

  Keysets, LU6.1-Sessions, LTACs, LTERM-Partner, Teilprogramme, Transaktionscodes und TAC-Queues können nur durch verzögertes Löschen aus der Konfiguration entfernt werden.

- *sofortiges Löschen* (KC_DELETE_OBJECT *subopcode1*=KC_IMMEDIATE)

  Sofortiges Löschen ist nur für Benutzerkennungen und LU6.1-Verbindungen von standalone UTM-Anwendungen erlaubt.

  Sofortiges Löschen aus der Konfiguration heißt, dass das Objekt und dessen Eigenschaften sofort aus der Objekttabelle gelöscht werden. Der Tabellenplatz einer „sofort gelöschten“ Benutzerkennung oder eines CON-Objekts kann direkt wieder von einer neu erzeugten Benutzerkennung bzw. einem neu erzeugten CON-Objekt belegt werden, ohne dass eine Neugenerierung der Anwendung durchgeführt werden muss. Der Name einer sofort gelöschten Benutzerkennung oder eines CON-Objekts bleibt **nicht** belegt. Es kann direkt nach dem Löschen eine neue Benutzerkennung bzw. ein neues CON-Objekt mit demselben Namen erzeugt werden.

  Auf ein so gelöschtes Objekt kann nicht mehr zugegriffen werden, weder lesend noch schreibend, auch nicht vom Administrator.

Pro KC_DELETE_OBJECT-Aufruf können Sie genau ein Objekt löschen (sofort oder verzögert). Innerhalb eines Teilprogramms können Sie mehrere KC_DELETE_OBJECT-Aufrufe hintereinander aufrufen, d.h. mehrere Objekte verschiedenen Typs löschen. Bei Objekten, die miteinander in Beziehung stehen, ist jedoch die Reihenfolge zu beachten, in der diese Objekte gelöscht werden. Ein Objekt, auf das sich andere Objekte beziehen, kann erst gelöscht werden, wenn diese anderen Objekte gelöscht sind bzw. die Beziehung aufgehoben wurde. Z.B. kann die Beziehung zwischen Terminal/Drucker und LTERM-Partner mit KDCSWTCH aufgehoben werden. Regeln, die Sie beim Löschen der Objekte beachten müssen, sind in den folgenden Abschnitten beschrieben.

Sowohl das sofortige als auch das verzögerte Löschen von Objekten erfolgt transaktionsgesichert. Das Objekt ist erst nach erfolgreichem Abschluss der Transaktion gelöscht, in der das KC_DELETE-OBJECT bearbeitet wird.

Gelöscht werden können nur Objekte, die in der Konfiguration enthalten sind. Ein Objekt, das Sie dynamisch in die Konfiguration eintragen, können Sie demnach erst löschen, wenn die Transaktion abgeschlossen ist, in der das Objekt eingetragen wird.

Das Löschen wirkt sowohl bei UTM-F- als auch bei UTM-S-Anwendungen über das Anwendungsende hinaus und kann nicht rückgängig gemacht werden.

UTM-Cluster-Anwendungen (Unix-, Linux- und Windows-Systeme)

In UTM-Cluster-Anwendungen gilt:
Der Aufruf wirkt Cluster-global, d.h. in allen Knoten-Anwendungen werden Objekte aus der Konfiguration gelöscht.
In UTM-Cluster-Anwendungen ist nur verzögertes Löschen erlaubt.

4.3.1 Clients/Drucker und LTERM-Partner löschen

Clients/Drucker und LTERM-Partner können nur verzögert aus der Konfiguration gelöscht werden.

Zum Löschen eines Client oder Druckers aus der Konfiguration müssen Sie KC_DELETE_OBJECT (mit *subopcode1*=KC_DELAY) für den Objekttyp KC_PTERM aufrufen. Zum Löschen eines LTERM-Partners müssen Sie den Objekttyp KC_LTERM angeben.

Einen Client/Drucker und den zugehörigen LTERM-Partner dürfen Sie nur löschen, wenn der Client/Drucker nicht mit der Anwendung verbunden ist. Deshalb sollten Sie den Client/Drucker vor dem Löschen sperren, um Fehler zu vermeiden. Das Sperren muss in einer eigenen Transaktion erfolgen. Zum Sperren des Client/Druckers siehe KDCPTERM im Abschnitt "KDCPTERM - Eigenschaften von Clients und Druckern ändern" oder KC_MODIFY_OBJECT mit *obj_type*=KC_PTERM im Abschnitt "obj_type=KC_PTERM".

Client/Drucker und der zugehörige LTERM-Partner stehen in Beziehung zueinander, deshalb muss beim Löschen von Clients, Druckern und ihrer LTERM-Partner auf die Reihenfolge geachtet werden. Allgemein gilt folgende Regel:

> Ein LTERM-Partner darf nicht gelöscht werden, solange ihm ein Client/Drucker zugeordnet ist.

Soll sowohl der Client/Drucker als auch der zugehörige LTERM-Partner aus der Konfiguration gelöscht werden, dann gilt die Regel:

> Client/Drucker (KC_PTERM) vor LTERM-Partner (KC_LTERM).

Beide Objekte können nur nacheinander in verschiedenen Transaktionen aus der Konfiguration gelöscht werden.

Beim Löschen von LTERM-Partnern ist Folgendes zu beachten:

- Bei UPIC-Clients (Typ UPIC-R und UPIC-L) und bei TS-Anwendungen (Typ APPLI oder SOCKET) müssen Sie vor dem Löschen des LTERM-Partners den Client aus der Konfiguration löschen.
- Bei Terminals und Druckern können Sie den LTERM-Partner löschen, ohne das Terminal bzw. den Drucker aus der Konfiguration herauszunehmen. In diesem Fall müssen Sie vor dem Löschen des LTERM-Partners den Client oder Drucker in einer eigenen Transaktion einem anderen LTERM-Partner zuordnen (KDCSWTCH im Abschnitt "KDCSWTCH - Zuordnung Clients, Drucker zu LTERM- Partnern ändern" oder KC_MODIFY_OBJECT mit *obj_type*=KC_PTERM im Abschnitt "obj_type=KC_PTERM").

Folgende LTERM- und PTERM-Partner dürfen Sie nicht löschen:

- LTERM-Partner, die zu einem LTERM-Pool gehören,
- LTERMs, die zu LTERM-Bündeln oder LTERM-Gruppen gehören,
- Druckersteuer-LTERMs,
- den LTERM-Partner KDCMSGLT, den openUTM intern für den MSGTAC-Service erzeugt,
- LTERM-Partner, die zu einem Multiplexanschluss gehören (nur auf BS2000-Systemen),
- LTERM- und PTERM-Partner, die in UTM-Cluster-Anwendungen (Unix-, Linux- und Windows-Systeme) für die Cluster-interne Kommunikation verwendet werden.

Alle anderen LTERM-Partner und Clients/Drucker können Sie unter Beibehaltung der obigen Regeln aus der Konfiguration löschen, unabhängig davon, ob sie statisch (mit KDCDEF) oder dynamisch in die Konfiguration eingetragen wurden.

> Den LTERM-Partner, der als Empfänger (*destadm*) für die Ergebnisse der asynchronen Administrationskommandos definiert ist, dürfen Sie löschen. Sie sollten jedoch in diesem Fall einen neuen Empfänger definieren, da sonst die Ergebnisse der asynchron bearbeiteten Administrationskommandos verloren gehen. Dazu stehen Ihnen der Aufruf KC_MODIFY_OBJECT mit Parametertyp KC_MAX_PAR und das Administrationskommando KDCAPPL zur Verfügung.

Das Löschen von Clients, Druckern und LTERM-Partnern hat folgende Auswirkungen:

- Zu einem gelöschten Client/Drucker kann keine Verbindung mehr aufgebaut werden. Damit kann nach dem Löschen keine Nachricht mehr an den Client bzw. Drucker gesendet werden.
- Es können keine Asynchron-Nachrichten, die an einen gelöschten LTERM-Partner gerichtet sind, mehr erzeugt werden, d.h. nach dem Löschen können keine Asynchron-Aufträge mehr in die Message Queue des LTERM-Partners eingetragen werden.
- Asynchron-Aufträge, die zum Zeitpunkt des Löschens in der Message Queue des LTERM-Partners stehen, also vor dem Löschen erzeugt wurden, können nicht mehr vom Client/Drucker aus der Queue gelesen werden. Die Asynchron-Aufträge in der Queue werden also nicht mehr bearbeitet. Sie sind jedoch noch administrierbar, d.h. sie können aus der Queue gelöscht werden. Dazu steht Ihnen der KDCS-Aufruf DADM zur Verfügung (siehe openUTM-Handbuch „Anwendungen programmieren mit KDCS“).
- Asynchron-Aufträge, die über einen inzwischen gelöschten LTERM-Partner erzeugt wurden, können noch ablaufen und sind administrierbar. Bei der Bearbeitung der Aufträge können aber keine weiteren Asynchron-Aufträge (Folge-Aufträge) mehr erzeugt werden.
- TLS-Bereiche (TLS = Terminal-spezifischer Langzeit-Speicher) eines gelöschten LTERM-Partners können noch gelesen und geschrieben werden.

4.3.2 Teilprogramme, Transaktionscodes und VORGANG-Exits löschen

Teilprogramme, Transaktionscodes, TAC-Queues und VORGANG-Exits können nur verzögert aus der Konfiguration gelöscht werden.

Zum Löschen eines Teilprogramms oder VORGANG-Exits aus der Konfiguration müssen Sie KC_DELETE_OBJECT (mit *subopcode1*=KC_DELAY) für den Objekttyp KC_PROGRAM aufrufen. Zum Löschen eines Transaktionscodes oder einer TAC-Queue müssen Sie den Objekttyp KC_TAC angeben.

Transaktionscodes und das Teilprogramm, dem diese Transaktionscodes zugeordnet sind, stehen in Beziehung zueinander. Ebenso steht ein VORGANG-Exit in Beziehung zu den Transaktionscodes, denen er zugeordnet ist. Deshalb müssen Sie beim Löschen von Transaktionscodes, Teilprogrammen und VORGANG-Exits auf die Reihenfolge achten. Allgemein gilt folgende Regel:

> Ein Teilprogramm/VORGANG-Exit darf erst gelöscht werden, nachdem alle zugehörigen Transaktionscodes gelöscht sind.

Folgende Teilprogramme dürfen nicht gelöscht werden:

- Teilprogramme, die zu den Event-Exits START, SHUT, FORMAT oder INPUT gehören.
- BS2000-Systeme: Teilprogramme und VORGANG-Exits, die in Lademodulen mit Lademodus STATIC gebunden sind.
- Unix, Linux- und Windows-Systeme: Teilprogramme und VORGANG-Exits, die statisch ins Anwendungsprogramm gebunden sind. D.h. Sie dürfen nur Teilprogramme und VORGANG-Exits löschen, die in Shared Objects bzw. DLLs enthalten sind.

Folgende Transaktionscodes dürfen nicht gelöscht werden:

- die Transaktionscodes KDCMSGTC, KDCSGNTC, KDCBADTC der Event-Services MSGTAC, SIGNON und BADTACS.
- das Administrationskommando KDCSHUT des Administrationsprogramms KDCADM.
- die von openUTM für XATMI intern erzeugten Transaktionscodes KDCTXCOM und KDCTXRLB.
- Transaktionscodes, die im Parameter SIGNON-TAC der BCAMAPPL-Anweisung definiert sind.

Folgende TAC-Queue darf nicht gelöscht werden:

- die Dead Letter Queue KDCDLETQ.

Alle anderen Teilprogramme und VORGANG-Exits, die nicht statisch gebunden sind, sowie Transaktionscodes können Sie aus der Konfiguration löschen, unabhängig davon, ob sie statisch oder dynamisch in die Konfiguration aufgenommen wurden.

> **i** Einen Asynchron-TAC bzw. eine TAC-Queue, der bzw. die als Empfänger (*destadm*) für die Ergebnisse der asynchronen Administrationskommandos definiert ist, dürfen Sie löschen. Sie sollten in diesem Fall jedoch einen neuen Empfänger definieren, da sonst die Ergebnisse verloren gehen. Dazu stehen Ihnen der Aufruf KC_MODIFY_OBJECT mit Parametertyp KC_MAX_PAR und das Administrationskommando KDCAPPL zur Verfügung.

Das Löschen von Teilprogrammen, VORGANG-Exits, Transaktionscodes und TAC-Queues hat folgende Auswirkungen:

- Gelöschte Teilprogramme und VORGANG-Exits werden nicht mehr aufgerufen.
- Asynchron-Aufträge an einen gelöschten Transaktionscode können nicht mehr erzeugt werden.
- Asynchron-Aufträge, die zum Zeitpunkt des Löschens in der Message Queue des Transaktionscodes stehen, also vor dem Löschen erzeugt wurden, werden nicht mehr bearbeitet. Die Aufträge bleiben in der Message Queue des Asynchron-TACs. Zur Entlastung des Pagepools sollten Sie die Asynchron-Aufträge aus der Queue löschen (siehe KDCS-Aufruf DADM im openUTM-Handbuch „Anwendungen programmieren mit KDCS“).
- Es können keine Dialog-Vorgänge an einen gelöschten TAC gestartet werden. Zum Zeitpunkt des Löschens offene Dialog-Vorgänge können noch normal abgewickelt werden, wenn nur der Vorgangs-TAC gelöscht wurde. Sie werden jedoch abgebrochen, wenn ein Folge-TAC aufgerufen wird, der gelöscht wurde.
- Beim Löschen einer TAC-Queue werden deren Nachrichten sofort gelöscht. Neue Nachrichten für eine gelöschte TAC-Queue können nicht erzeugt werden.

4.3.3 Benutzerkennungen löschen

Eine Benutzerkennung können Sie „verzögert“ oder „sofort“ aus der Konfiguration löschen (siehe "Objekte dynamisch aus der Konfiguration löschen"). In UTM-Cluster-Anwendungen (Unix-, Linux- und Windows-Systeme) ist nur verzögertes Löschen möglich.

Zum Löschen einer Benutzerkennung aus der Konfiguration müssen Sie KC_DELETE_OBJECT (mit *subopcode1* =KC_DELAY oder KC_IMMEDIATE) für den Objekttyp KC_USER aufrufen.

Sie können bis auf die untenstehenden Ausnahmen alle Benutzerkennungen löschen, die explizit in die Konfiguration eingetragen wurden (statisch oder dynamisch).

Folgende Benutzerkennungen können nicht gelöscht werden:

- Die Benutzerkennung KDCMSGUS, die openUTM intern für den MSGTAC-Service erzeugt
- Benutzerkennungen, die einem Terminal für ein automatisches KDCSIGN zugeordnet sind (siehe "Clients, Drucker und LTERM-Partner eintragen").
- Verbindungsbenutzerkennungen; d.h. Benutzerkennungen, die einem Client vom Typ UPIC, APPLI oder SOCKET fest zugeordnet sind.

In Anwendungen ohne explizit generierte Benutzerkennungen ist das Löschen von intern erzeugten Benutzerkennungen generell nicht möglich.

Beim Löschen von Benutzerkennungen bestehen die folgenden Einschränkungen bzgl. des Zeitpunkts, zu dem eine Benutzerkennung gelöscht werden kann:

Eine Benutzerkennung können Sie nur löschen (verzögert oder sofort), wenn zum Zeitpunkt des Löschens kein Benutzer oder Client mit dieser Benutzerkennung bei der Anwendung angemeldet ist. Deshalb sollten Sie die Benutzerkennung vor dem Löschen sperren, um Fehler zu vermeiden. Das Sperren muss in einer eigenen Transaktion erfolgen. Zum Sperren einer Benutzerkennung siehe KDCUSER im Abschnitt "KDCUSER - Benutzereigenschaften ändern" oder KC_MODIFY_OBJECT mit *obj_type*=KC_USER im Abschnitt "obj_type=KC_USER".

Das sofortige Löschen einer Benutzerkennung ist außerdem zeitweise nicht möglich, wenn:

- zum Zeitpunkt des Löschens unter dieser Benutzerkennung ein Asynchron-Auftrag bearbeitet wird, d.h. aus der Message Queue geholt und gestartet wird.
- zum Zeitpunkt des Löschens für die Benutzerkennung eine verteilte Transaktion im PTC-Zustand ist (PTC=Prepare to Commit).
- wenn der User-spezifische Langzeit-Speicher (ULS) der Benutzerkennung nicht gesperrt werden kann, z.B. weil gerade ein Administrator oder Administrationsprogramm auf den ULS zugreift.

Verzögertes Löschen

Das verzögerte Löschen von Benutzerkennungen hat folgende Auswirkungen:

- Mit einer verzögert gelöschten Benutzerkennung kann sich kein Benutzer/Client mehr bei der Anwendung anmelden.
- Asynchron-Vorgänge, die vor dem Löschen von der Benutzerkennung erzeugt wurden und zum Zeitpunkt des Löschens noch nicht bearbeitet sind, können noch ablaufen und sind administrierbar. Diese Vorgänge können aber selbst keine Asynchron-Aufträge mehr erzeugen.
- Ein offener Dialog-Vorgang kann nicht mehr fortgesetzt werden. Gesicherte Vorgangsdaten eines Benutzers (z. B. LSSB-Daten, Dialognachricht) werden gelöscht:

  - in stand-alone-Anwendungen beim nächsten Anwendungsstart
  - in UTM-Cluster-Anwendungen (Unix-, Linux- und Windows-Systeme) beim nächsten Start der Knoten-Anwendung, an der der Benutzer als letztes angemeldet war

  Die Daten werden nicht gelöscht, wenn ein offener Vorgang eine Transaktion im Zustand PTC hat. In diesem Fall muss zunächst die Transaktion vor- oder zurückgesetzt werden. Eine Transaktion im Zustand PTC können Sie z.B. per Programmschnittstelle zurücksetzen (Operationscode KC_PTC_TA).
- ULS-Bereiche (ULS = User-spezifischer Langzeit-Speicher) einer gelöschten Benutzerkennung können noch gelesen und geschrieben werden.
- Alle Nachrichten in der Message Queue zu dieser Benutzerkennung werden sofort gelöscht. Es können keine neuen Nachrichten für diese Message Queue erzeugt werden.

Sofortiges Löschen

Das sofortige Löschen einer Benutzerkennung hat folgende Auswirkungen:

- Mit einer sofort gelöschten Benutzerkennung kann sich kein Benutzer/Client mehr bei der Anwendung anmelden.
- Asynchron-Aufträge, die vor dem Löschen von der Benutzerkennung erzeugt und von openUTM in der Message Queue zwischengespeichert wurden, laufen nicht mehr an, d.h. sie werden von openUTM nicht mehr bearbeitet. Die Aufträge werden in dem Moment gelöscht, in dem openUTM sie aus der Message Queue abholt, weil sie bearbeitet werden sollen.

  Wenn Sie mit DADM RQ (siehe "Informieren über Nachrichten in einer Queue - DADM RQ") die Informationen über Aufträge in der Message Queue abfragen, dann gibt openUTM für die Aufträge einer gelöschten Benutzerkennung statt der Auftrag gebenden Benutzerkennung den Wert *NONE aus.
- Aufträge an LTERM- oder LPAP-Partner, die von der Benutzerkennung gestartet wurden und noch in der Message Queue des Partners stehen, werden noch gesendet.
- Ein offener Dialog-Vorgang, der von einer gelöschten Benutzerkennung gestartet wurde, wird ebenfalls sofort gelöscht. Offene Dialog-Vorgänge für einen nicht angemeldeten Benutzer können z.B. dann existieren, wenn sich der Benutzer innerhalb eines Vorgangs, in dem schon einen Sicherungspunkt erreicht wurde, mit KDCOFF abgemeldet hat.
- Auf ULS-Bereiche (ULS = User-spezifischer Langzeit-Speicher) einer gelöschten Benutzerkennung kann nicht mehr zugegriffen werden. Sie werden gelöscht.
- Alle Nachrichten in der Message Queue zu dieser Benutzerkennung werden sofort gelöscht.

4.3.4 Keysets löschen

Keysets können nur verzögert aus der Konfiguration gelöscht werden. Zum Löschen eines Keysets müssen Sie KC_DELETE_OBJECT (mit *subopcode1*=KC_DELAY) für den Objekttyp KC_KSET aufrufen.

Einschränkung: Das Keyset KDCAPLKS kann nicht gelöscht werden.

Objekte, die ein gelöschtes Keyset referenzieren, verlieren ihre Zugriffsrechte. TACs, TAC-Queues und Benutzerkennungen können jedoch dynamisch andere Keysets zugeordnet werden.

4.3.5 LU6.1 - Verbindungen und Sessions löschen

Zum Löschen einer LU6.1-Transportverbindung zwischen der lokalen UTM-Anwendung und einer Partner-Anwendung müssen Sie KC_DELETE_OBJECT (in stand-alone-Anwendungen mit *subopcode1*=KC_IMMEDIATE, in UTM-Cluster-Anwendungen mit KC_DELAY) für den Objekttyp KC_CON aufrufen. Wenn Sie eine LU6.1-Session löschen wollen, rufen Sie KC_DELETE_OBJECT (mit *subopcode1*=KC_DELAY) für den Objekttyp KC_LSES auf.

Löschen von LU6.1-Verbindungen

Das Löschen eines CON-Objekts ist nicht möglich, wenn es mit der Anwendung verbunden ist.

Besonderheiten beim Löschen von LU6.1-Sessions

Ein LSES-Objekt (LU6.1-Session) kann nur gelöscht werden, wenn

- die Session nicht aufgebaut ist und
- sich keine der beiden Half-Sessions im Zustand PTC befindet.

Um zu prüfen, ob sich eine Session im PTC-Zustand befindet, können Sie den Zustand der Session abfragen (z.B. mit KC_GET_OBJECT mit Objekttyp LSES).

Zum Löschen eines LSES-Objektes wird folgendes Vorgehen empfohlen:

1. Bauen Sie vor dem Löschen des Objekts zunächst die Session auf.
2. Setzen Sie die Session dann auf 'Quiet'.
3. Löschen Sie nach dem Verbindungsabbau das Objekt mit dem oben genannten Aufruf.

4.3.6 LTACs löschen

Transaktionscodes, über die Service-Programme in Partner-Anwendungen gestartet werden, können nur verzögert aus der Konfiguration gelöscht werden.

Zum Löschen eines LTACs müssen Sie KC_DELETE_OBJECT (mit *subop-code1*=KC_DELAY) für den Objekttyp KC_LTAC aufrufen.

4.4 Objekteigenschaften ändern

Mit dem Aufruf KC_MODIFY_OBJECT können Sie während eines Anwendungslaufs die Eigenschaften von Objekten und Parameter des Anwendungsprogramms ändern und Aktionen veranlassen (z.B. Rücksetzen von Statistikwerten).

KC_MODIFY_OBJECT, siehe "KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern"

Folgende Objekttypen haben dynamisch veränderbare Eigenschaften:

KC_CLUSTER_NODE, KC_DB_INFO, KC_KSET, KC_LOAD_MODULE, KC_LPAP, KC_LSES, KC_LTAC, KC_LTERM, KC_MUX, KC_OSI_CON, KC_OSI_LPAP, KC_PTERM, KC_TAC, KC_TACCLASS, KC_TPOOL, KC_USER.

In den folgenden Abschnitten wird auf das Modifizieren einiger Objekttypen näher eingegangen (KC_PTERM, KC_LTERM, KC_TAC, KC_USER, KC_KSET und KC_LSES).

Folgende Parametertypen haben Eigenschaften, die dynamisch verändert werden können:

KC_CLUSTER_CURR_PAR, KC_CLUSTER_PAR, KC_CURR_PAR, KC_DIA-G_AND_ACCOUNT_PAR, KC_MAX_PAR, KC_TASKS_PAR, KC_TIMER_PAR.

Pro KC_MODIFY_OBJECT-Aufruf können Sie genau ein Objekt ändern. Es ist jedoch möglich innerhalb eines Administrationsprogramms KC_MODIFY_OBJECT mehrfach aufzurufen, um die Eigenschaften mehrerer Objekte zu modifizieren. Beim Aufruf geben Sie den Typ des Objektes, seinen Namen und die Eigenschaften an, die geändert werden sollen.

Bei der Änderung von Anwendungsparametern können Sie innerhalb eines Aufrufs alle Parameter ändern, die zu demselben Parametertyp gehören.

Welche Eigenschaften bei welchem Objekttyp oder Anwendungsparameter geändert werden können und welche Aktionen dadurch ausgelöst werden, ist im Abschnitt „KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern“ beschrieben.

Die Wirksamkeit und Dauer einer Änderung ist abhängig vom Objekttyp bzw. Anwendungsparameter und der Eigenschaft, die verändert wird. Es gibt Änderungen, die nur im aktuellen Anwendungslauf wirksam sind (Run) und Änderungen, die über den aktuellen Anwendungslauf hinaus wirken (Durable). Der Zeitpunkt, zu dem eine Änderung wirksam wird, kann sein:

- sofort (Immediate),
- nach der Transaktionssicherung (PEND),
- wenn es die Auslastung der Anwendung zulässt.

UTM-Cluster-Anwendungen (Unix-, Linux- und Windows-Systeme)

In einer UTM-Cluster-Anwendung gilt:
Je nach Objekt kann der Aufruf sowohl Aktionen, die Cluster-global als auch solche, die Knoten-lokal wirken, anstoßen. Global wirkende Aktionen werden für jede Knoten-Anwendung der UTM-Cluster-Anwendung ausgeführt, unabhängig davon, ob eine Knoten-Anwendung gerade aktiv ist oder nicht. Lokal wirkende Aktionen wirken sich nur auf die Knoten-Anwendung aus, an der sie ausgeführt wurden.

4.4.1 Clients/Drucker und LTERM-Partner modifizieren

Zum Ändern der Eigenschaften eines Client oder Druckers müssen Sie KC_MODIFY_-OBJECT mit Objekttyp KC_PTERM aufrufen. Zum Ändern der Eigenschaften eines LTERM-Partners müssen Sie den Objekttyp KC_LTERM angeben.

LTERM-Partner, die zu einem LTERM-Pool gehören, oder Clients/Drucker, die sich über einen LTERM-Pool anschließen, können nicht modifiziert werden.

Bei Clients/Druckern und LTERM-Partnern können Sie den Status und den aktuellen Zustand der Verbindung zum Client/Drucker verändern. Eine Änderung des Status (sperren/freigeben) wirkt nach der Transaktionssicherung über den Anwendungslauf hinaus. Eine Änderung des aktuellen Zustands (Verbindung ist vorhanden, ist nicht vorhanden, befindet sich im Aufbau) wirkt dann, wenn es die Auslastung der Anwendung zulässt, aber nicht über den Anwendungslauf hinaus.

Wenn Sie die Zuordnung Client/Drucker zu einem LTERM-Partner ändern wollen, muss der Partner existieren und darf nicht gelöscht sein. Der LTERM-Partner darf nicht für den Anschluss an einen Client vom Type UPIC konfiguriert sein. Außerdem darf der LTERM-Partner kein Master oder Slave eines LTERM-Bündels und kein Alias- oder Primary-LTERM einer LTERM-Gruppe sein. Eine Änderung der Zuordnung wirkt nach der Transaktionssicherung über den Anwendungslauf hinaus.

Bei Clients/Druckern darf entweder nur der eventuell zugeordnete LTERM-Partner oder nur ein Modus für den automatischen Verbindungsaufbau beim Start der Anwendung geändert werden. Das Anfordern eines automatischen Verbindungsaufbaus beim Start der Anwendung ist nur möglich, wenn der Client/Drucker nicht gesperrt ist. Eine Änderung für den Verbindungsaufbau beim Start wirkt nach der Transaktionssicherung über den Anwendungslauf hinaus.

Hinweis für BS2000-Systeme:

Ist der LTERM-Partner einem Terminal zugeordnet, können Sie die Formatattribute verändern. Ein spezifisches Startformat ist jedoch nur nutzbar für Anwendungen ohne Benutzerkennungen oder wenn ein eigener Anmelde-Vorgang definiert ist. Eine Änderung der Formatattribute wirkt nach der Transaktionssicherung über den Anwendungslauf hinaus.

4.4.2 Transaktionscodes und TAC-Queues modifizieren

Zum Ändern der Eigenschaften eines TACs oder einer TAC-Queue müssen Sie KC_MODIFY_OBJECT mit dem Objekttyp KC_TAC aufrufen.

Es ist nicht möglich, den Status eines TACs zu ändern und gleichzeitig spezifische Statistikwerte zurückzusetzen.

Änderungen am Status eines TACs oder einer TAC-Queue wirken sofort und über den Anwendungslauf hinaus. Änderungen an den Statistikwerten eines Transaktionscodes wirken sofort.

Wenn Sie Zugriffe auf einen Transaktionscode über ein Keyset steuern wollen, können Sie der Access Liste des Transaktionscodes ein existierendes Keyset zuweisen. Dann müssen Sie jedoch einen eventuell vorhandenen Lockcode entfernen (auf Null setzen). Umgekehrt, wenn der Zugriff auf den Transaktionscode durch einen Lockcode geschützt wird, darf in der Access Liste kein Keyset definiert sein.

Eine TAC-Queue können Sie ebenfalls mit einem Keyset gegen unbefugtes Lesen/Löschen und Schreiben schützen. Dazu weisen Sie den Parametern *q_read_acl* und/oder *q_write_acl* jeweils das gewünschte Keyset zu.

Änderungen an den Parametern, die Zugriffe regeln, wirken nach der Transaktionssicherung über den Anwendungslauf hinaus.

Für Asynchron-Transaktionscodes mit CALL=BOTH/FIRST und TAC-Queues kann die Sicherung von Nachrichten in der Dead Letter Queue bei fehlerhafter Verarbeitung ein- und ausgeschaltet werden. Für MSGTAC und KDCDLETQ ist diese Sicherung nicht möglich. Das Ein- und Ausschalten der Sicherung in die Dead Letter Queue wirkt nach Transaktionsende über den Anwendungslauf hinaus.

4.4.3 Benutzerkennungen modifizieren

Zum Ändern der Eigenschaften einer Benutzerkennung oder der zugeordneten USER-Queue müssen Sie KC_MODIFY_OBJECT mit dem Objekttyp KC_USER aufrufen.

Benutzerkennungen mit Administrationsberechtigung können Sie nicht sperren. Ebenfalls nicht möglich ist es, Eigenschaften von Benutzerkennungen zu modifizieren, die einem Client vom Typ APPLI, SOCKET oder UPIC zugeordnet sind.

Wenn Sie das Passwort für eine Benutzerkennung ändern wollen, achten Sie darauf, dass

- das neue Passwort der Komplexitätsstufe entspricht, die für die Benutzerkennung definiert ist,
- nicht das bisherige Passwort wieder verwendet wird, wenn für die Benutzerkennung nur Passwörter mit begrenzter Gültigkeitsdauer zulässig sind,

Eine Benutzerkennung können Sie mit Zugriffsrechten (Keyset) versehen oder diese ändern.

Eine USER-Queue können Sie mit einem Keyset vor unbefugtem Lesen/Löschen und Schreiben schützen. Dazu weisen Sie den Parametern *q_read_acl* und/oder *q_write_acl* jeweils das gewünschte Keyset zu (siehe "kc_user_str, kc_user_fix_str, kc_user_dyn1_str bzw. kc_user_dyn2_str - Benutzerkennungen").

Alle Änderungen, die Sie an den Eigenschaften einer Benutzerkennung oder einer USER-Queue vornehmen, wirken nach der Transaktionssicherung über den Anwendungslauf hinaus.

4.4.4 Keysets modifizieren

Zum Ändern der Keys eines Keysets müssen Sie KC_MODIFY_OBJECT mit dem Objekttyp KC_KSET aufrufen.

Beachten Sie dabei, dass das Keyset KDCAPLKS nicht verändert werden kann und die Angabe eines Keys kleiner als 1 oder größer als der in der Anwendung erlaubte Maximalwert (KDCDEF-Anweisung MAX, Operand KEYVALUE) nicht zulässig ist.

Keysets mit dem Attribut MASTER können ebenfalls nicht verändert werden.

4.4.5 LU6.1 - Sessions modifizieren

Zum Ändern der Eigenschaften einer LU6.1 Session müssen Sie KC_MODIFY_OBJECT mit dem Objekttyp KC_LSES aufrufen.

Für eine LU6.1-Session können Sie einen Verbindungsaufbau bzw. Verbindungsabbau veranlassen und bei einem Verbindungsaufbau eine Transportverbindung zur Session zuordnen.

Wenn Sie einen sofortigen Verbindungsaufbau anfordern, darf die Eigenschaft QUIET nicht gesetzt und der LPAP-Partner nicht gesperrt sein. Wenn Sie den sofortigen Verbindungsabbau anfordern, darf keine der anderen Eigenschaften verändert werden.

Bei Angabe einer Transportverbindung zur Session sollten Sie sicherstellen, dass die Verbindung existiert und für den zugehörige LPAP-Partner generiert ist.

Alle Änderungen, die Sie an einem LSES-Objekt vornehmen, wirken nur, wenn es die Auslastung der Anwendung zulässt.

5 Generierungsanweisungen aus der KDCFILE erzeugen

openUTM stellt Ihnen den inversen KDCDEF zur Verfügung, mit dem Sie aus den aktuellen Konfigurationsdaten in der KDCFILE Steueranweisungen für das UTM-Tool KDCDEF erzeugen können. So gehen Änderungen in der Konfiguration, die Sie während des Betriebs der Anwendung vorgenommen haben, bei einer Neugenerierung Ihrer Anwendung nicht verloren.

KDCDEF-Steueranweisungen, die der inverse KDCDEF erzeugt

Der inverse KDCDEF erzeugt Steueranweisungen für die Objekttypen, für die das dynamische Eintragen und Löschen möglich ist. Steueranweisungen für andere Objekte und Komponenten der Anwendung sowie für Anwendungsparameter erzeugt der inverse KDCDEF nicht. Mit dem inversen KDCDEF können Sie also folgende KDCDEF-Steueranweisungen erzeugen:

- USER-Anweisungen

  für alle aktuell in der Anwendung existierenden Benutzerkennungen. Der inverse KDCDEF erstellt keine USER-Anweisungen für die von openUTM intern erzeugten Benutzerkennungen.

  In Anwendungen ohne Benutzerkennungen erstellt der inverse KDCDEF keine USER-Anweisungen.
- LTERM-Anweisungen

  für alle LTERM-Partner der Anwendung, die nicht zu einem LTERM-Pool oder zu einem Multiplexanschluss gehören.
- PTERM-Anweisungen

  für alle Clients und Drucker, die in der Konfiguration eingetragen sind. Für Clients, die zu einem LTERM-Pool oder zu einem Multiplexanschluss gehören, werden keine PTERM-Anweisungen erzeugt.
- PROGRAM-Anweisungen

  für alle Teilprogramme und Exits, die aktuell in der Konfiguration der Anwendung enthalten sind.
- TAC-Anweisungen

  für alle Transaktionscodes und TAC-Queues der Anwendung.
- KSET-Anweisungen

  für alle Keysets der Anwendung.
- CON-Anweisungen

  für alle LU6.1-Verbindungen der Anwendung.
- LSES-Anweisungen

  für alle LU6.1-Sessions der Anwendung.
- LTAC-Anweisungen

  für alle Transaktionscodes für Partner-Anwendungen.

Der inverse KDCDEF erzeugt Steueranweisungen für alle Objekte der Anwendung, die zu einem dieser Objekttypen gehören unabhängig davon, ob die Objekte dynamisch in die Konfiguration eingetragen oder statisch bei einer vorherigen KDCDEF-Generierung erzeugt wurden. Dabei werden alle Modifikationen berücksichtigt, die Sie für diese Objekte während des Anwendungslaufs vorgenommen haben.

Der inverse KDCDEF erzeugt **keine** Steueranweisungen für Objekte, die dynamisch aus der Konfiguration der Anwendung gelöscht wurden. Diese Objekte sind somit nach der folgenden Neugenerierung endgültig aus der Konfiguration gelöscht. Sie belegen dann keinen Tabellenplatz mehr und die Namen der Objekte können bei der Neugenerierung bereits wieder verwendet werden.

Darüber hinaus überträgt das UTM-Tool KDCUPD nach der Neugenerierung mit KDCDEF keine Anwendungsdaten zu den dynamisch gelöschten Objekten aus der alten KDCFILE in die neue KDCFILE. Auch dann nicht, wenn in der neuen KDCDEF-Generierung ein Objekt mit Namen und Objekttyp eines gelöschten Objektes vorhanden ist. Insbesondere werden von KDCUPD keine Asynchron-Aufträge übertragen, die über inzwischen gelöschte LTERM-Partner oder Benutzerkennungen erzeugt wurden.

Die vom inversen KDCDEF erzeugten USER-Anweisungen enthalten keine Passwörter. Für Benutzerkennungen, die mit Passwort generiert sind, erzeugt der inverse KDCDEF USER-Steueranweisungen der Form:

`USER` *`name`*`, PASS=*RANDOM,....`

Nach dem Erzeugen einer neuen KDCFILE, d.h. nach dem folgenden KDCDEF-Lauf, müssen Sie die Passwörter der Benutzerkennungen mit dem UTM-Tool KDCUPD in die neue KDCFILE übertragen (siehe openUTM-Handbuch „Anwendungen generieren“). Das ist auch bei einer UTM-F-Anwendung möglich.

> **i** Für eine UTM-Cluster-Anwendung sind die Passwörter in der Cluster-User-Datei enthalten und müssen nicht mit KDCUPD in eine neue KDCFILE übertragen werden.

5.1 Starten des inversen KDCDEF

Sie können den inversen KDCDEF „online“ oder „offline“ starten. „Online“ heißt, Sie starten den inversen KDCDEF, während die Anwendung läuft. „Offline“ heißt, Sie starten den inversen KDCDEF nach Beendigung des Anwendungslaufs.

In beiden Fällen können Sie den inversen KDCDEF so aufrufen, dass er für alle möglichen Objekte KDCDEF-Steueranweisungen produziert. Sie können den inversen KDCDEF aber auch so aufrufen, dass er nur Steueranweisungen für bestimmte Objekttypen erzeugt, die in den Objektgruppen CON, DEVICE, KSET, LSES, LTAC, PROGRAM und USER zusammengefasst sind.

Sie können KDCDEF-Steueranweisungen für nur eine oder mehrere dieser Gruppen anfordern.

Inversen KDCDEF online starten

Um einen inversen KDCDEF-Lauf online starten zu können, müssen Sie ein eigenes Anwendungsprogramm erstellen, das KC_CREATE_STATEMENTS aufruft.

KC_CREATE_STATEMENTS, siehe "KC_CREATE_STATEMENTS - KDCDEF-Steueranweisungen erzeugen (inverser KDCDEF)"

Wann der inverse KDCDEF-Lauf gestartet wird, ist abhängig davon, ob zum Zeitpunkt des KC_CREATE_STATEMENTS-Aufrufs ein anderer Prozess schreibend auf die Konfigurationsdaten zugreift. Folgende Fälle sind zu unterscheiden:

- Zum Zeitpunkt des KC_CREATE_STATEMENTS-Aufrufs laufen Transaktionen, die die Konfigurationsdaten der Anwendung modifizieren, Passwörter oder Locales ändern. In diesem Fall wird durch den KC_CREATE_STATEMENTS-Aufruf ein Asynchron-Auftrag erzeugt. Der inverse KDCDEF-Lauf wird erst gestartet, wenn diese Transaktionen abgeschlossen sind.
  Neue Transaktionen dieser Art können jedoch solange nicht gestartet werden, bis der inverse KDCDEF-Lauf abgeschlossen, d.h. der Asynchron-Auftrag bearbeitet ist.

  In UTM-Cluster-Anwendungen gilt außerdem:
  Eine Cluster-global wirkende Administrationsaktion führt in jeder laufenden Knoten-Anwendung zu einer Transaktion, die den Start des inversen KDCDEF verzögern kann. Umgekehrt kann die Ausführung einer globalen Administrationsaktion auf einem laufenden Knoten verzögert werden, wenn dort gerade ein inverser KDCDEF läuft.
- Zum Zeitpunkt des KC_CREATE_STATEMENTS-Aufrufs laufen

  keine Transaktionen, die die Konfigurationsdaten, Passwörter oder Locales ändern.
  In diesem Fall wird der inverse KDCDEF-Lauf sofort (synchron) gestartet. Er ist bereits bei der Rückkehr in das Teilprogramm beendet. D.h. zu diesem Zeitpunkt sind bereits alle angeforderten KDCDEF-Steueranweisungen erzeugt und in Dateien abgelegt.

Hinweis zu UTM-Cluster-Anwendungen:
Ein online inverser KDCDEF kann nicht gestartet werden, solange in einer UTM-Cluster-Anwendung unterschiedlich generierte Knoten-Anwendungen laufen.

Ein inverser KDCDEF-Lauf unterliegt nicht der Transaktionssicherung.

Mit Hilfe des online ausgeführten inversen KDCDEF können Sie parallel zum Anwendungslauf alle Vorbereitungen für die Neugenerierung Ihrer Anwendung treffen. Die Ausfallzeit wird dadurch minimiert.

Sie können den inversen KDCDEF auch über die Administrationstools WinAdmin und WebAdmin online starten.

Inversen KDCDEF offline starten

Sie starten den inversen KDCDEF offline, d.h. außerhalb des Anwendungsbetriebs, indem Sie das UTM-Generierungstool KDCDEF aufrufen und die Steueranweisung CREATE-CONTROL-STATEMENTS absetzen.

CREATE-CONTROL-STATEMENTS siehe openUTM-Handbuch „Anwendungen generieren“

Die vom inversen KDCDEF erzeugten Dateien können dann im selben oder in einem späteren KDCDEF-Lauf verarbeitet werden.

5.2 Ergebnis des inversen KDCDEF-Laufs

Der inverse KDCDEF schreibt die Steueranweisungen entweder alle in eine Datei oder er schreibt die Steueranweisungen jeder Objektgruppe in eine eigene Datei.

Auf BS2000-Systemen können die Steueranweisungen statt in eine Datei auch in ein LMS-Bibliothekselement geschrieben werden.

Die vom inversen KDCDEF geschriebenen Dateien können Sie bei der Neugenerierung der Anwendung als Input an den KDCDEF übergeben. Dazu geben Sie für jede dieser Dateien die Steueranweisung OPTION DATA=*filename* an.

Die Dateien, die der inverse KDCDEF erzeugt, können Sie direkt als Input-Dateien an den KDCDEF übergeben. Sie können die Dateien jedoch auch editieren, d.h. vor dem folgenden KDCDEF-Lauf modifizieren.

Bei LMS-Bibliothekselementen auf BS2000-Systemen ist es von deren Typ abhängig, ob sie modifizierbar sind oder nicht; nur textartige Elemente (Elemente vom Typ S, M, J, P, D oder X) sind modifizierbar.

Die Namen der Dateien, die der inverse KDCDEF erzeugt, legen Sie beim Start des inversen KDCDEF fest. Existiert keine Datei dieses Namens, dann wird sie automatisch angelegt. Existiert eine Datei dieses Namens, dann können Sie festlegen, ob sie überschrieben oder fortgeschrieben werden soll.

5.3 Inverser KDCDEF bei Versionsübergängen

Bei einem Übergang auf eine neue openUTM-Version müssen Sie die KDCDEF-Steueranweisungen zunächst in der Vorgängerversion erzeugen. Sie müssen dazu den inversen KDCDEF der Vorgängerversion starten. Die von diesem KDCDEF erzeugten Dateien können Sie als Input-Dateien für den KDCDEF der neuen openUTM-Version verwenden.

5.4 Empfehlungen für die Neugenerierung einer Anwendung

Beim Betrieb einer UTM-Anwendung kann es unumgänglich werden, die Anwendung neu zu generieren, d.h. erneut einen KDCDEF-Lauf durchzuführen. Mögliche Gründe können sein:

- Die bei der Generierung festgelegten Maximalwerte müssen angepasst werden.
- Für die verteilte Verarbeitung über LU6.1 oder OSI TP müssen möglicherweise neue Objekte erzeugt werden, weil sich z.B. der Server-Verbund bei der verteilten Verarbeitung vergrößern soll.

  Für die verteilte Verarbeitung über LU6.1 ist ein KDCDEF-Lauf nur dann erforderlich, wenn neue LPAP-Objekte eingefügt werden müssen. Objekte vom Typ CON, LSES und LTAC lassen sich dagegen auch durch dynamische Administration erzeugen (vorausgesetzt es wurden ausreichend Tabellenplätze mit der RESERVE-Anweisung freigehalten).
- Neue Lademodule, Shared Objects oder DLLs müssen in das Anwendungsprogramm eingefügt werden.
- Die reservierten Tabellenplätze für das dynamische Eintragen von Objekten in die Konfiguration sind belegt. Die Tabellen müssen erweitert oder zum Löschen vorgemerkte Objekte müssen endgültig gelöscht werden, um die Tabellenplätze freizugeben.

Die Ausfallzeit Ihrer Anwendung, die eine solche Neugenerierung mit sich bringt, können Sie minimieren. Beachten Sie dazu die folgenden **Empfehlungen**:

- Bereits bei der Erstgenerierung Ihrer Anwendung sollten Sie die Steueranweisungen für den KDCDEF auf mehrere Dateien verteilen, die Sie KDCDEF dann mit OPTION DATA=zur Verfügung stellen. Insbesondere sollten Sie die Steueranweisungen USER, LTERM, PTERM, PROGRAM, TAC, CON, KSET, LSES und LTAC in separate Dateien schreiben. Dabei sollten die Anweisungen, die zu einer Gruppe gehören (siehe "Starten des inversen KDCDEF"), in eine Datei geschrieben werden. So können Sie diese Dateien bei einer späteren Neugenerierung der Anwendung durch die vom inversen KDCDEF erzeugten Dateien ersetzen.
- Vor einer Neugenerierung der Anwendung und vor dem inversen KDCDEF-Lauf sollten Sie alle Objekte dynamisch aus der Konfiguration löschen (KC_DELETE_OBJECT), die in der neuen Konfiguration nicht mehr enthalten sein sollen. Das dynamische Löschen hat gegenüber dem manuellen Löschen der zugehörigen Steueranweisungen aus der Input-Datei für den KDCDEF folgende Vorteile:
  - Das manuelle Löschen von KDCDEF-Anweisungen aus der KDCDEF-Input-Datei ist unkomfortabel und fehleranfällig. Es muss beim Löschen auf Abhängigkeiten zwischen den Objekten und damit zwischen den KDCDEF-Anweisungen geachtet werden. Werden Abhängigkeiten übersehen, muss der KDCDEF-Lauf wiederholt werden. Die Ausfallzeit wird damit vergrößert.
  - Die Abläufe bei der Neugenerierung können automatisiert werden durch Aufruf des offline inversen KDCDEF mit anschließendem KDCUPD, siehe openUTM-Handbuch „Anwendungen generieren“.

Beim manuellen Löschen eines Objekts können u.U. Daten, die zu den gelöschten Objekten in der KDCFILE gespeichert sind, vom KDCUPD, der im Zusammenhang mit der folgende Neugenerierung durchgeführt wird, in die neue KDCFILE übernommen werden. Dabei ist folgender Fall zu beachten:

Sie wollen für ein bestimmtes Objekt verhindern, dass KDCUPD die Daten aus der alten KDCFILE überträgt, die zu diesem Objekt gehören (z.B. weil das „neue“ Objekt zwar den gleichen Namen und Typ, aber andere Eigenschaften hat). Sie können beim KDCUPD aber nur die Datenübernahme für alle Objekte eines bestimmten Objekttyps ausschließen, nicht aber die Übernahme der Daten für ein bestimmtes Objekt. Sie sollten daher das Objekt dynamisch aus der Konfiguration löschen. In der Neugenerierung sollte das Objekt wieder enthalten sein.

In diesem Fall überträgt KDCUPD die Daten nicht, die zu diesem Objekt gehören, da KDCUPD Daten von gelöschten Objekten nicht überträgt.

Änderungsgenerierung einer UTM-Cluster-Anwendung siehe entsprechendes Unterkapitel im openUTM-Handbuch „Einsatz von UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen“.

Beispiel

In der neuen Konfiguration soll ein Transaktionscode enthalten sein mit dem Namen eines Asynchron-Transaktionscodes, der in der „alten“ Konfiguration existierte. Der neue Transaktionscode ruft jedoch einen anderen Service auf (wird einem anderen Teilprogramm zugeordnet). Es sind folgende Fälle zu unterscheiden:

- Die Eigenschaften des „alten“ Transactionscodes wurden geändert:
  In diesem Fall überträgt KDCUPD, wenn Sie TRANSFER ASYNTACS=YES angeben, die Message Queue des „alten“ Transaktionscodes zusammen mit den in der Queue enthaltenen Asynchron-Aufträgen in die neue KDCFILE und ordnet sie dem „neuen“ Transaktionscode zu.
  KDCUPD mit TRANSFER ASYNTACS=NO bewirkt, dass für keinen Asynchron-Transaktionscode die Message Queue aus der alten in die neue KDCFILE übertragen wird.
- Der „alte“ Transaktionscode wurde dynamisch aus der Konfiguration gelöscht, in der neuen Konfiguration ist er wieder enthalten:
  In diesem Fall überträgt KDCUPD, auch wenn Sie TRANSFER ASYNTACS=YES angeben, die Message Queue des „alten“ Transaktionscodes nicht in die neue KDCFILE, weil KDCUPD keine Daten von gelöschten Objekten überträgt.

Entsprechendes gilt für die Message Queues von LTERM-Partnern und USER Queues von Benutzern.

6 Administration über Kommandos

Die Administrationskommandos von openUTM können Sie nur nutzen, wenn folgende Voraussetzungen erfüllt sind:

- Das Standard-Administrationsprogramm KDCADM muss generiert sein (KDCDEF-Anweisung PROGRAM) bzw. dynamisch in die Konfiguration eingetragen werden (Administrationsprogramm mit KC_CREATE_OBJECT und *obj_type*=KC_PROGRAM).
- Die Administrationskommandos, die Sie nutzen wollen, müssen als Transaktionscodes generiert sein (KDCDEF-Anweisung TAC) bzw. dynamisch in die Konfiguration eingetragen werden (Administrationsprogramm mit KC_CREATE_OBJECT und *obj_type*=KC_TAC).

Zur KDCDEF-Generierung der Kommandos und der für das Aufrufen der Kommandos erforderlichen Berechtigung siehe Kapitel „Zugriffsrechte und Zugriffsschutz".

An der Kommandoschnittstelle von openUTM steht für jede Administrationsfunktion von KDCADM sowohl ein Dialog- als auch ein Asynchron-Kommando zur Verfügung. Sie können also alle Aktionen (Ausnahme: Anwendungslauf abbrechen mit KDCSHUT KILL) sowohl im Dialog als auch über Message Queuing ausführen.

> **i** Die Kommandos von openUTM können sowohl von Benutzern am Terminal als auch von Client-Programmen und Partner-Anwendungen verwendet werden. Sie sind jedoch in erster Linie für die Eingabe am Terminal gedacht. Zur Administration durch Client-Programme und andere Anwendungen ist die Programmschnittstelle zur Administration wesentlich besser geeignet.

6.1 Administration im Dialog

Die Dialog-Administrationskommandos können genutzt werden von:

- Benutzern an Terminals
- UPIC-Clients
- TS-Anwendungen
- HTTP-Clients
- LU6.1- oder OSI TP-Partner-Anwendungen
- anderen Dialog-Teilprogrammen der Anwendung

> ¡ Die Benutzer, LPAPs und OSI-LPAPs, die die Kommandos aufrufen, müssen Administrationsberechtigung haben.

Eingabe der Administrationskommandos

Ein Benutzer am Terminal muss die Kommandos im Linemode eingeben. Eingaben über Formate werden nicht akzeptiert (Ausnahme: Kommandos, die keine Operanden haben).

Der Vorteil der Kommandoeingabe im Linemode liegt darin, dass die Kommandobearbeitung weniger Zeit erfordert und die Administrationsaufgaben auch vermischt mit anderen Vorgängen durchgeführt werden können.

> ¡ In UTM-Anwendungen auf BS2000-Systemen werden Administrationskommandos abgewiesen, wenn bei der letzten Ausgabe ein Edit-Profil verwendet wurde.

Ausgabe der Ergebnisse

Das Ergebnis der Kommandobearbeitung liefert openUTM an den Auftraggeber zurück. Die Ausgabe am Terminal erfolgt ebenfalls im Linemode.

Passt bei der Ausgabe am Terminal eine Ausgabe nicht auf eine Bildschirmseite, so bietet openUTM i.A. in der letzten Bildschirm-Zeile ein Fortsetzungskommando an, mit dem die Ausgabe an der aktuellen Position fortgesetzt werden kann.

Wie die Ergebnismeldungen zu den einzelnen Kommandos aussehen, ist im Kapitel „Administrationskommandos - KDCADM" bei der Beschreibung des jeweiligen Kommandos dargestellt.

Die Ausgabe nach erfolgreicher Bearbeitung des Administrationskommandos besagt nicht unbedingt, dass die Aktion erfolgreich durchgeführt wurde, die Sie angefordert haben. Bei einigen Kommandos bedeutet die Meldung lediglich, dass openUTM die Aktion angestoßen hat (z.B. Verbindungsaufbau, Programmaustausch). Der Grund dafür ist, dass sich die Durchführung dieser Aktionen über einen längeren Zeitraum erstreckt oder dass openUTM die Aktion erst zu einem späteren Zeitpunkt durchführen kann. Ob die entsprechende Aktion erfolgreich ausgeführt werden konnte, können Sie über eine spätere Abfrage mit KDCINF ermitteln. Bei einigen dieser Aktionen (z.B. Programmaustausch) erzeugt openUTM nach Abschluss der Bearbeitung K-Meldungen, denen Sie entnehmen können, ob die Aktion erfolgreich durchgeführt wurde oder nicht. Diese Meldungen gehen z.B. standardmäßig an das Meldungsziel SYSLOG und werden auf der Standardausgabe (SYSOUT/stderr) ausgegeben.

6.2 Administration über Message Queuing

Die Asynchron-Kommandos können aufgerufen werden von:

- Benutzern an Terminals
- TS-Anwendungen
- LU6.1- oder OSI TP-Partner-Anwendungen
- anderen Dialog- oder Asynchron-Teilprogrammen der Anwendung

> ¡ Die Benutzer/(OSI-)LPAPs, die die Kommandos aufrufen, müssen Administrationsberechtigung haben.

Durch das Absetzen eines Asynchron-Kommandos wird ein Asynchron-Auftrag erzeugt, den openUTM in die Message Queue des zugehörigen Administrations-TACs von KDCADM einreiht. Der Auftrag wird dann entkoppelt vom Auftraggeber oder Teilprogramm ausgeführt.

Die Asynchron-Kommandos ermöglichen eine „programmierte bzw. automatische Administration“. Die vom Standard-Administrationsprogramm KDCADM gelieferten Daten können dabei an ein anderes Teilprogramm übergeben werden, das die Daten auswertet und entsprechende Aktionen (Aufruf weiterer Kommandos oder Transaktionscodes) einleitet. Die Asynchron-Kommandos können z.B. auch von dem Event-Service MSGTAC aufgerufen werden, der mit dem Aufruf eines Administrationskommandos auf bestimmte Ereignisse (UTM-Meldungen) reagiert.

Absetzen der Administrationskommandos

Am Terminal müssen die Asynchron-Kommandos wie bei der Administration im Dialog im Linemode eingegeben werden. Partner-Anwendungen übergeben die Kommandos zusammen mit den Operanden als Asynchron-Nachrichten an die Anwendung. Es werden dieselben Operanden wie im Dialog übergeben. Die Asynchron-Kommandos unterscheiden sich nur durch ihre Namen von den Dialog-Kommandos.

Ein KDCS-Teilprogramm ruft ein Asynchron-Kommando auf, indem es entweder einen FPUT NE-Aufruf absetzt oder einen DPUT NE-Aufruf, wenn das Kommando zu einem bestimmten Zeitpunkt ausgeführt werden soll.

Das KDCS-Parameterfeld KCRN des Aufrufs versorgen Sie mit dem Namen des Asynchron-Kommandos (=Transaktionscode). Der Nachrichtenbereich des Aufrufs muss die Operandenliste des Administrationskommandos enthalten. Jedes Administrationskommando müssen Sie in einem FPUT- bzw. DPUT-Aufruf übergeben.

Mehrere Aufrufe des gleichen Administrationskommandos, die in einer Transaktion bearbeitet werden sollen, können Sie als Teilnachrichten senden. Jede Teilnachricht muss ein Administrationskommando (einschließlich der Operanden) enthalten. Das Administrationsprogramm KDCADM liest die Teilnachrichten in einer Schleife von FGET-Aufrufen und verarbeitet sie.

`FPUT NT` oder `DPUT NT`	Erster Aufruf des Administrationskommandos, z.B. KDCLTRMA
`FPUT NT` oder `DPUT NT`	Zweiter Aufruf von KDCLTRMA
...	Weitere Aufrufe von KDCLTRMA
`FPUT NE` oder `DPUT NE`	Letzter Aufruf von KDCLTRMA

Die Benutzerkennung, unter der das Teilprogramm läuft, muss Administrationsberechtigung haben. Das MSGTAC-Teilprogramm hat immer Administrationsberechtigung (siehe auch openUTM-Handbuch „Anwendungen programmieren mit KDCS“, MSGTAC-Teilprogramm).

Ausgabe des Ergebnisses

Nach der Bearbeitung des Auftrags informiert openUTM durch eine Asynchron-Nachricht über das Ergebnis. Die Nachricht hat folgendes Format:

Kopfzeile
1. Ergebniszeile (= 1. Bildschirmzeile wie bei Dialog-Ausgabe)
2. Ergebniszeile (= 2. Bildschirmzeile wie bei Dialog-Ausgabe)

:

:

Es werden soviele Ergebniszeilen ausgegeben wie beim entsprechenden Dialog-Kommando. Lediglich die für die Blätterfunktion im Dialog ausgegebene Zeile entfällt.

Wie die Bildschirmzeilen der Dialog-Ausgabe aufgebaut sind, ist im Kapitel „Administrationskommandos - KDCADM" bei der Beschreibung des jeweiligen Kommandos dargestellt.

Aufbau der Kopfzeile

ADMCMD:	Kommando-Name	Leerzeichen	Operanden des Administrationskommandos
8 Bytes	8 Bytes	1 Byte	... variabel ...

Empfänger des Ergebnisses

Alle Nachrichten, die von den Asynchron-Kommandos erzeugt werden, gehen an denselben Empfänger. Der Empfänger (DESTADM) kann entweder bei der KDCDEF-Generierung oder im laufenden Betrieb durch die Administration festgelegt werden, entweder mit WinAdmin, WebAdmin oder über die Programmschnittstelle KDCADMI (*opcode*=KC_MODIFY_OBJECT und *object_type*=KC_MAX_PAR, siehe im Abschnitt "obj_type = KC_MAX_PAR"). Es kann jederzeit durch die Administration ein anderer Empfänger definiert werden. Als Empfänger kann ein weiterer Asynchron-TAC, der das Ergebnis weiter verarbeitet, oder der LTERM-Partner eines Terminals, Druckers oder einer TS-Anwendung angegeben werden.

Ist kein Empfänger definiert, dann führt openUTM die Administrationskommandos zwar aus, aber die Ergebnis-Nachrichten gehen verloren.

Ist jedoch ein Asynchron-TAC als Empfänger definiert und ist dieser nicht verfügbar, z.B. weil er gesperrt ist, dann wird das Kommando nicht ausgeführt und openUTM erzeugt die Meldung K076.

Ist der Empfänger ein LTERM-Partner, dann wird das Ergebnis als Asynchron-Nachricht ausgegeben.
Ist der Empfänger ein Asynchron-TAC, dann muss das zugehörige Teilprogramm jede einzelne Zeile des Ergebnisses mit einem FGET-Aufruf lesen. Der erste FGET-Aufruf liefert die Kopfzeile, jeder weitere eine Bildschirmzeile.

> ¡ Das Layout der Ausgaben unterliegt nicht der Kompatibilitätsgarantie, d.h. es kann sich beim Übergang auf eine neue openUTM-Version ändern. Teilprogramme, die die Ausgaben der Administrationskommandos auswerten, müssen beim Versionsübergang deshalb eventuell angepasst werden.

Zuordnung zwischen Auftrag und Ergebnis beim Empfänger

Den Operanden eines Asynchron-Kommandos können Sie einen Kommentar in Anführungszeichen ("kommentar") mitgeben. Dieser Kommentar kann dann vom Empfänger der Ergebnis-Nachricht ausgewertet werden.

Als Kommentar können Sie z.B. eine Auftragsnummer angeben. Durch diese Auftragsnummer kann der Empfänger den Auftrag identifizieren.

Der Kommentar sollte dann vor den Operanden stehen, damit die Auftragsidentifikation immer am Nachrichtenanfang steht und gut zu adressieren ist.

```
Asynchron-Kommando "kommentar" operanden
```

7 Erstellen eigener Administrationsprogramme

Mit Hilfe der Programmschnittstelle KDCADMI können Sie eigene Administrationsprogramme erstellen. Ein Administrationsprogramm müssen Sie immer als KDCS-Teilprogramm schreiben, d.h. es muss durch einen INIT und einen PEND-Aufruf eingerahmt sein. Der PEND-Aufruf sollte immer die Transaktion beenden.

Sie können Administrationsprogramme erstellen:

- als Dialog-Teilprogramme für die Administration im Dialog
- als Asynchron-Teilprogramme für die Administration über Message Queues, z.B. für eine automatische Administration, siehe Kapitel „Administration automatisieren".

Jedes Administrationsprogramm besitzt folgenden Aufbau:

```
INIT
...
MGET (oder FGET, falls Asynchron-Programm)
... Eingabe analysieren
KDCADMI  (Administrationsschnittstelle aufrufen)
[KDCADMI] (ggf. mehrere Aufrufe)
...

[RSET]

MPUT (oder FPUT/DPUT)
PEND
```

Sie können innerhalb eines Administrationsprogramms mehrere Administrationsaufrufe absetzen. Wenn Sie innerhalb einer Transaktion mehrere Aufrufe starten, dann müssen Sie jedoch beachten, dass einige Aufrufe in einer bestimmten Reihenfolge kommen müssen und dass eine Reihe der über Administrationsprogramme angestoßenen Aktionen der Transaktionssicherung unterliegen, d.h. sie werden erst nach erfolgreichem PEND-Aufruf ausgeführt. In diesem Fall sollten Sie einen RSET-Aufruf für den Fehlerfall vorsehen.

Eine UTM-Anwendung kann mehrere Administrationsprogramme für unterschiedliche Zwecke besitzen. Ein Administrationsprogramm kann von einem Terminal, einem Client, einem anderen Teilprogramm (z.B. MSGTAC) oder einer anderen Anwendung aus gestartet werden.

7.1 Dialog-Administrationsprogramme

Wenn Sie im Dialog administrieren wollen, dann können Sie:

- mehrere Administrationsaufgaben in ein Programm zusammenfassen oder
- die Administration als Mehrschritt-Vorgang programmieren und
- die Daten über Formate ein- und ausgeben (nur auf BS2000-Systemen)

Die beiden folgenden Beispiele skizzieren, wie Sie dies umsetzen können.

7.1.1 Mehrere Administrationsaufrufe

In diesem Beispiel soll ein in mehreren Versionen vorhandenes Lademodul, Shared Object bzw. eine DLL im laufenden Betrieb gegen eine neue Version ausgetauscht und um ein neues Teilprogramm mit einem neuen TAC erweitert werden. Der Austausch verläuft in drei Schritten.

Zuerst müssen über KDCADMI einige Daten abgefragt werden wie z.B. die aktuelle geladene Version des Lademoduls/ des Shared Objects/der DLL, bevor im zweiten Schritt die Konfiguration (TAC- und PROGRAM-Anweisung) geändert wird. In einem letzten Schritt wird der eigentliche Austausch veranlasst.

```
#include <kcadminc.h>        /* Include-Datei fuer die Administration    */
INIT
...
MGET                         /* Daten (Name, TAC,...) des zu tauschenden */
                             /* Teilprogramms einlesen                   */
... Eingabe analysieren
/******************** 1. Teil: Pruefen und Abfragen *********************/
KDCADMI opcode=KC_GET_OBJECT /* Ist noch Platz fuer TAC- PROGRAM,...     */
                             /* -Anweisungen reserviert ?                */

KDCADMI opcode=KC_GET_OBJECT /* Pruefen, ob es die TAC-, PROGRAM- ...    */
                             /* Anweisungen schon gibt                   */
KDCADMI opcode=KC_GET_OBJECT /* Aktuelle Version des Lademoduls / Shared */
                             /* Objects ermitteln                        */
if {Fehler in Teil 1:
    MPUT mit PEND FI }       /* Bei Fehler Meldung an Bildschirm         */
```

```
/******************** 2. Teil: Dynamisch generieren *****************/
KDCADMI opcode=KC_CREATE_OBJECT
                             /* PROGRAM-Anweisung einfuegen             */
KDCADMI opcode=KC_CREATE_OBJECT
                             /* TAC-Anweisung einfuegen                 */
if {Fehler in Teil 2: RSET}  /* Bei Fehler Transaktion zurücksetzen     */
```

```
/******************** 3. Teil: Programm austauschen *******************/
KDCADMI opcode=KC_MODIFY_OBJECT
                              /* Teilprogramm austauschen               */

MPUT                          /* Meldung an Bildschirm                  */
PEND FI
```

Der RSET-Aufruf ist notwendig, damit keine inkonsistente Generierung entsteht, wenn in Teil 2 Fehler auftreten. Die KC_CREATE_OBJECT-Operationen müssen für die aufgeführten Objekte in dieser Reihenfolge (PROGRAM-TAC) angegeben werden, sonst kann openUTM die Bezüge nicht herstellen.

7.1.2 Mehrschritt-Vorgang

In diesem Beispiel werden im ersten Schritt Informationen über die UTM-Anwendung eingeholt und, falls erforderlich, in einem zweiten Schritt Objekteigenschaften modifiziert. Die beiden Programme arbeiten mit einem #Format.

```
/*********************** Teilprogramm ADMREAD ***************************/
#include <kcadminc.h>        /* Include-Datei fuer die Administration    */
INIT

MGET ... KCMF=#FORMADM       /* Eingaben werden ueber ein Format         */
                             /* eingelesen, die Eingabe wird analysiert  */
KDCADMI opcode=KC_GET_OBJECT
                             /* Administationsaufruf, openUTM liefert    */
                             /* die Daten an das Programm                */
MPUT KCMF=#FORMADM           /* Daten/Ergebnis an Bildschirm ausgeben    */

PEND RE KCRN=ADMMOD          /* Vorgang wird fortgesetzt                 */
```

```
/*********************** Teilprogramm ADMMOD ****************************/
#include <kcadminc.h>       /* Include-Datei fuer die Administration     */
INIT

MGET ... KCMF=#FORMADM      /* Eingaben werden ueber ein Format          */
                            /* eingelesen, die Eingabe wird analysiert   */
KDCADMI opcode=KC_MODIFY_OBJECT
                            /* Das gewuenschte Objekt wird modifiziert   */
                            /* Es sind mehrere KDCADMI-Aufrufe moeglich  */
MPUT KCMF=#FORMADM          /* Daten/Ergebnis an Bildschirm ausgeben     */

PEND FI                     /* Vorgang wird beendet                      */
```

Sie können die Programme z.B. wie folgt ausbauen:

- Sie analysieren die Rückgaben des KDCADMI-Aufrufs und geben bei Fehlern mit Hilfe einer Fehlerroutine eine entsprechende Nachricht aus
- oder Sie schreiben in ADMREAD die gelieferten Daten in einen LSSB, der in ADMMOD wiederverwendet werden kann.

openUTM auf Unix-, Linux- und Windows-Systemen unterstützt kein Formatierungssystem. Wenn Sie das Programm über den *utmdtp* aus einer Shell bzw. einem DOS-Fenster aufrufen wollen, müssen Sie deshalb die MGET- und MPUT-Aufrufe im Zeilenmodus programmieren.

Dieses Programm können Sie auch über einen UPIC-Client ansprechen.

7.2 Diagnosemöglichkeiten für die Administrationsschnittstelle

Zur Fehlerdiagnose für die Aufrufe an die Administrationsschnittstelle gibt es die beiden Bereiche Administration DIAGAREA und Administration USERAREA im UTM-Dump sowie den ADMI-Trace als eigene Datei. Im einzelnen bietet openUTM folgende Diagnosemöglichkeiten:

- In der UTM Diagarea zeigt der KDCS-Opcode ADMI den Aufruf der Administrationsschnittstelle an.
- In der Administration DIAGAREA werden alle Aufrufe mitprotokolliert.
  Die Administration DIAGAREA ist analog zur UTM Diagarea aufgebaut und wird zyklisch beschrieben.
- In der Administration USERAREA werden Prozess-spezifisch die an openUTM übergebenen Daten (Datenbereich oder Selektionsbereich) mitprotokolliert.
  Die Administration USERAREA enthält jeweils nur die Daten eines Aufrufes an die Administrationsschnittstelle.
- Zur Diagnose von Fehlern in Teilprogrammen, die die Programmschnittstelle zur Administration (KDCADMI) verwenden, können Sie den ADMI-Trace einschalten.
- Auf BS2000-Systemen werden bei eingeschalteter SAT-Protokollierung und bei Auswahl des UTM-Ereignisses ADM-CMD alle Aufrufe an die Administrationsschnittstelle aufgezeichnet. Zusätzlich werden bei opcode=KC_GET_OBJECT die Returncodes KC_MC_OK und KC_MC_LAST_ELT als erfolgreich mitprotokolliert.

Die Beschreibung der Administration DIAGAREA, der Administration USERAREA und des ADMI-Trace sowie den Aufbau der SAT-Protokollsätze finden Sie im zur jeweiligen Plattform gehörigen openUTM-Handbuch „Meldungen, Test und Diagnose“.

8 Zentrale Administration mehrerer Anwendungen

Wenn Sie mehrere UTM-Anwendungen zentral administrieren wollen, dann können Sie entweder WinAdmin oder WebAdmin einsetzen oder die Administration über selbst erstellte Kommandoprozeduren oder Administrationsprogramme durchführen.

- Mit **WinAdmin** und **WebAdmin** stehen Ihnen über eine komfortable Bedienoberfläche alle Funktionen der Programmschnittstelle zur Verfügung. Sie können gleichzeitig mehrere UTM-Anwendungen administrieren, die auf unterschiedlichen Rechnern mit BS2000-, Unix-, Linux- oder Windows-Systemen ablaufen.

  WinAdmin und WebAdmin lassen sich leicht und schnell einsetzen, da Sie nichts programmieren müssen, weder auf dem Administrationsrechner noch auf den zu administrierenden UTM-Anwendungen.

- Sie können eigene Kommandoprozeduren oder Programme erstellen, wenn Sie z.B. Funktionen nutzen möchten, die nicht von WinAdmin oder WebAdmin angeboten werden.

  Die Administrationsaufgaben werden aufgeteilt in einen zentralen Teil, die Administrationsanwendung, und einen dezentralen Teil, der auf der jeweils zu administrierenden UTM-Anwendung läuft.

  Die zentrale Administration können Sie sowohl mit Hilfe der Kommandoschnittstelle als auch mit Hilfe der Programmschnittstelle abwickeln. Es wird empfohlen, für die Administration von Anwendungen immer die Programmschnittstelle einzusetzen.

  Für die Konfiguration der zentralen Administration stehen Ihnen mehrere Grundmodelle zur Verfügung, siehe Abschnitt "Konfigurationsmodelle für eigene Administrationsanwendungen".

Administration von UTM-Cluster-Anwendungen (Unix-, Linux- und Windows-Systeme)

Sie können die Knoten-Anwendungen einer UTM-Cluster-Anwendung gemeinsam administrieren.

- Mit **WinAdmin** und **WebAdmin** stehen Ihnen Administrationsfunktionen zur Verfügung, die Sie auf alle Knoten-Anwendungen der UTM-Cluster-Anwendung global anwenden können. Außerdem bieten WinAdmin und WebAdmin z.B. auch zusammenfassende Statistikanzeigen, die alle laufenden Knoten-Anwendungen einbeziehen.

  Aus diesem Grund wird empfohlen, UTM-Cluster-Anwendungen mit WinAdmin oder WebAdmin zu administrieren.

- Sie können wie gewohnt eigene Kommandoprozeduren oder Programme erstellen. Für die Administration von UTM-Cluster-Anwendungen werden zusätzliche Datenstrukturen angeboten:
  - Für den Parametertyp KC_CLUSTER_PAR ist die Datenstruktur *kc_cluster_par_str* definiert. In *kc_cluster_par_str* liefert UTM die aktuellen Einstellungen für die globalen Eigenschaften einer UTM-Cluster-Anwendung und aktuelle Daten (z.B. Generierungs-, Startzeitpunkt, Anzahl der aktiven und der generierten Knoten-Anwendungen) zurück (siehe Abschnitt „kc_cluster_par_str - Globale Eigenschaften einer UTM-Cluster-Anwendung").
  - Für den Objekttyp KC_CLUSTER_NODE ist die Datenstruktur *kc_cluster_node_str* definiert. In *kc_cluster_node_str* liefert UTM die Eigenschaften der einzelnen Knoten-Anwendungen (Instanzen) einer UTM-Cluster-Anwendung zurück (siehe Abschnitt „kc_cluster_node_str - Knoten-Anwendungen einer UTM-Cluster-Anwendung").
  - Für den Objekttyp KC_CLUSTER_CURR_PAR ist die Datenstruktur *kc_cluster_curr_par_str* definiert. In *kc_cluster_curr_par_str* liefert UTM aktuelle Werte der UTM-Cluster-Anwendung zurück (siehe Abschnitt "kc_cluster_curr_par_str - Statistikwerte einer UTM-Cluster-Anwendung"). Außerdem lassen sich mit *kc_cluster_curr_par_str* Statistikzähler der UTM-Cluster-Anwendung zurücksetzen.

Im Abschnitt "Administration über UPIC-Client" finden Sie ein Generierungsbeispiel für die Administration einer UTM-Cluster-Anwendung über einen UPIC-Client.

Weitere Informationen zur Administration von UTM-Cluster-Anwendungen finden Sie im openUTM-Handbuch „Einsatz von UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen“

8.1 Administration über WinAdmin und WebAdmin

In diesem Abschnitt erhalten Sie einen Einstieg in das Arbeiten mit WinAdmin und WebAdmin. Ausführliche Informationen finden Sie

- in der **WinAdmin-Beschreibung**, die einen umfassenden Überblick über den Funktionsumfang und die Bedienung von WinAdmin bietet. Dieses Dokument ist online als PDF-Datei verfügbar.
- in der **WebAdmin-Beschreibung**, die einen umfassenden Überblick über den Funktionsumfang und die Bedienung von WebAdmin bietet. Dieses Dokument ist online als PDF-Datei verfügbar.
- im **Online-Hilfesystem**, das kontextsensitiv alle Dialogfelder und die zugehörigen Parameter beschreibt, die die grafische Oberfläche von WinAdmin bzw. WebAdmin bietet. Außerdem wird dargestellt, wie man WinAdmin bzw. WebAdmin konfiguriert, um UTM-Anwendungen administrieren zu können.

Mit WinAdmin und WebAdmin können Sie den kompletten Funktionsumfang der KDCADMI nutzen und z.B. Objekte dynamisch in eine Konfiguration aufnehmen, Objekte löschen oder UTM-Anwendungen starten und beenden. Außerdem stehen zusätzliche Funktionen zur Verfügung, die nicht über KDCADMI zugänglich sind:

- Definition von Meldungskollektoren, um UTM-Meldungen von den laufenden UTM-Anwendungen abzufragen, anzuzeigen und zu archivieren
- Administration von Message Queues
- Druckeradministration und Druckersteuerung
- Anzeigen von GSSB-Inhalten und Löschen von GSSBs
- Erzeugen und Löschen von Temporären Queues
- Zusammenfassen mehrerer Administrationsschritte in einer Transaktion (nur WinAdmin)
- Sehr weitgehende Unterstützung des UTM-Securitykonzeptes über Rollen und Access Lists
- Definition von Aktionen, z.B. das Speichern von Statistikwerten in Dateien oder Reaktionen auf Schwellwertüberschreitungen bzw. -unterschreitungen
- Sammeln und Archivieren von Statistikdaten der UTM-Anwendungen.

Aus der Sicht von openUTM sind WinAdmin und WebAdmin Clients vom Typ UPIC-R. Bevor Sie eine UTM-Anwendung über WinAdmin oder WebAdmin administrieren können, müssen Sie deshalb

- in der UTM-Anwendung den Zugriff durch WinAdmin bzw. WebAdmin generieren (siehe "Generierung der UTM-Anwendung anpassen")
- und in WinAdmin bzw. WebAdmin die Verbindungsparameter konfigurieren (siehe "WinAdmin und WebAdmin konfigurieren").

8.1.1 Generierung der UTM-Anwendung anpassen

Auf der Seite der UTM-Anwendung muss der Zugriff auf das Programm KDCWADMI sowie die UPIC-Anbindung von WinAdmin bzw. WebAdmin generiert werden.

Zugriff auf die Programmschnittstelle ermöglichen

Um den Zugriff auf die Programmschnittstelle zu ermöglichen, müssen das Programm KDCWADMI und der TAC KDCWADMI generiert werden. Dazu sind folgende KDCDEF-Anweisungen notwendig:

```
PROGRAM KDCWADMI, COMP=ILCS                    BS2000-Systeme

PROGRAM KDCWADMI, COMP=C                       Unix-, Linux- und Windows-Systeme
```

und

```
TAC KDCWADMI, PROGRAM=KDCWADMI, CALL=BOTH,
    ADMIN=Y
```

Das Teilprogramm KDCWADMI wird mit openUTM ausgeliefert und kann zur Anwendung gebunden oder von der Anwendung nachgeladen werden.

Zum Starten von UTM-Anwendungen mit WinAdmin oder WebAdmin oder zum Anstoßen von KDCDEF-/KDCUPD-Läufen mit WinAdmin muss openFT installiert und konfiguriert sein. Das Senden oder Holen von Dateien kann WinAdmin auch mittels FTP durchführen.

WinAdmin und WebAdmin als UPIC-Client bekannt machen

Zusätzlich muss in allen über WinAdmin oder WebAdmin zu administrierenden UTM-Anwendungen WinAdmin bzw. WebAdmin als UPIC-Client generiert werden.
Als Beispiel dienen folgende KDCDEF-Anweisungen (PTERM/LTERM):

```
        BCAMAPPL bcamappl_name,                  BS2000-Systeme
                 T-PROT=RFC1006

        BCAMAPPL bcamappl_name,                  Unix-, Linux- und Windows-Systeme
                 T-PROT=RFC1006,                 Hinweis: LISTENER-PORT ist zwar kein
                 LISTENER-PORT= port             Pflicht-Parameter, aber in der Praxis notwendig.

und     PTERM pterm-name,
              LTERM= lterm-name,
              BCAMAPPL= bcamappl-name,
              PRONAM= processor-name,
              PTYPE=UPIC-R

        LTERM lterm-name

        MAX PRIVILEGED-LTERM= lterm-name

        USER wadmin, PASS=C'XYZ',
             PERMIT=ADMIN,
             RESTART=NO

oder    TPOOL LTERM= upiclt,                     Hinweis: In diesem Fall entfällt allerdings die
              NUMBER=10,                          Möglichkeit, diese Verbindung als
              PRONAM=*ANY,                        privilegiertes LTERM einzurichten.
              PTYPE=UPIC-R,
              BCAMAPPL= bcamappl-name
```

Die Bezeichnungen *pterm-name*, *lterm-name*, *bcamappl-name, upiclt* und *wadmin* sind unter Berücksichtigung der Namenskonventionen frei wählbar. *pterm-name* ist der Name, den Sie dem WinAdmin- bzw. WebAdmin-Client geben. *bcamappl-name* ist der Name, den Sie der Anwendung für die Client/Server-Kommunikation geben. *upiclt* ist das Präfix für den Namen des LTERM-Partners. *wadmin* ist eine administrationsberechtigte Benutzerkennung für die Anwendung. *XYZ* bezeichnet das Passwort zur Benutzerkennung *wadmin*.

Die Vergabe eines Passworts ist nicht zwingend, es sollte aber aus Gründen der Anwendungssicherheit immer ein Passwort verwendet werden.

Den hier vergebenen Anwendungsnamen, die Benutzerkennung und ggf. das Passwort benötigen Sie bei der Konfiguration von WinAdmin bzw. WebAdmin.

8.1.2 WinAdmin und WebAdmin konfigurieren

Beim ersten Start von WinAdmin bzw. WebAdmin wird eine Konfigurationsdatenbank eingerichtet. In dieser Datenbank müssen zunächst die Verwaltungsdaten der UTM-Anwendungen gespeichert werden, die über WinAdmin oder WebAdmin administriert werden sollen. Mit diesen Daten legen Sie WinAdmin- bzw. WebAdmin-seitig fest,

- wie die Anwendung heißt,
- auf welchem Rechner sie abläuft,
- wie die Verbindung beschaffen ist und
- welche Benutzer diese Anwendung administrieren dürfen.

Diese Daten werden den WinAdmin- bzw. WebAdmin-Objekten „Hosts“, „UTM-Anwendungen“, „UPIC-Verbindungen “ und „WinAdmin-Benutzer" bzw. „WebAdmin-Benutzer“ zugeordnet.

Außerdem können Sie Kollektionen definieren. Eine Kollektion enthält eine oder mehrere UTM-Anwendungen. Standardmäßig ist die Kollektion <Alle UTM-Anwendungen> eingerichtet.

> **i** Beim Wechsel der WinAdmin- bzw. WebAdmin-Version können Sie die Konfigurationsdatenbank der Vorgängerversion verwenden.

Konfiguration der WinAdmin- und WebAdmin-Objekte

In der folgenden Tabelle sind die WinAdmin- und WebAdmin-Objekte aufgelistet, die definiert sein müssen.

Objekt	Beschreibung und Eigenschaften
Hosts	Dieses Objekt bezeichnet innerhalb von WinAdmin oder WebAdmin den Rechner, auf dem die UTM-Anwendung läuft (Anwendungsrechner).
UTM-Anwendungen	Dieses Objekt bezeichnet die zu administrierende UTM-Anwendung.
UPIC-Verbindungen	Mit diesem Objekt definieren Sie die Verbindung von WinAdmin oder WebAdmin zu der Anwendung.
WinAdmin-/WebAdmin-Benutzer	Nach der Installation gibt es nur die WinAdmin-/WebAdmin-Benutzerkennung „Master“, die zu allem berechtigt ist. Es wird empfohlen, weitere Benutzerkennungen mit eingeschränkten Rechten zu definieren.
Kollektionen	Dieses Objekt fasst UTM-Anwendungen zu einer Kollektion zusammen.

Details siehe WinAdmin- bzw. WebAdmin-Beschreibung.

Arbeiten mit Kollektionen

Ein WinAdmin-/WebAdmin-Benutzer kann mehrere Anwendungen zu einer Kollektion zusammenfassen, um diese einfacher administrieren zu können.

Mit WinAdmin ist es sogar möglich, Objekte verschiedener Anwendungen einer geöffneten Kollektion gemeinsam, d. h. in einem Schritt zu administrieren.

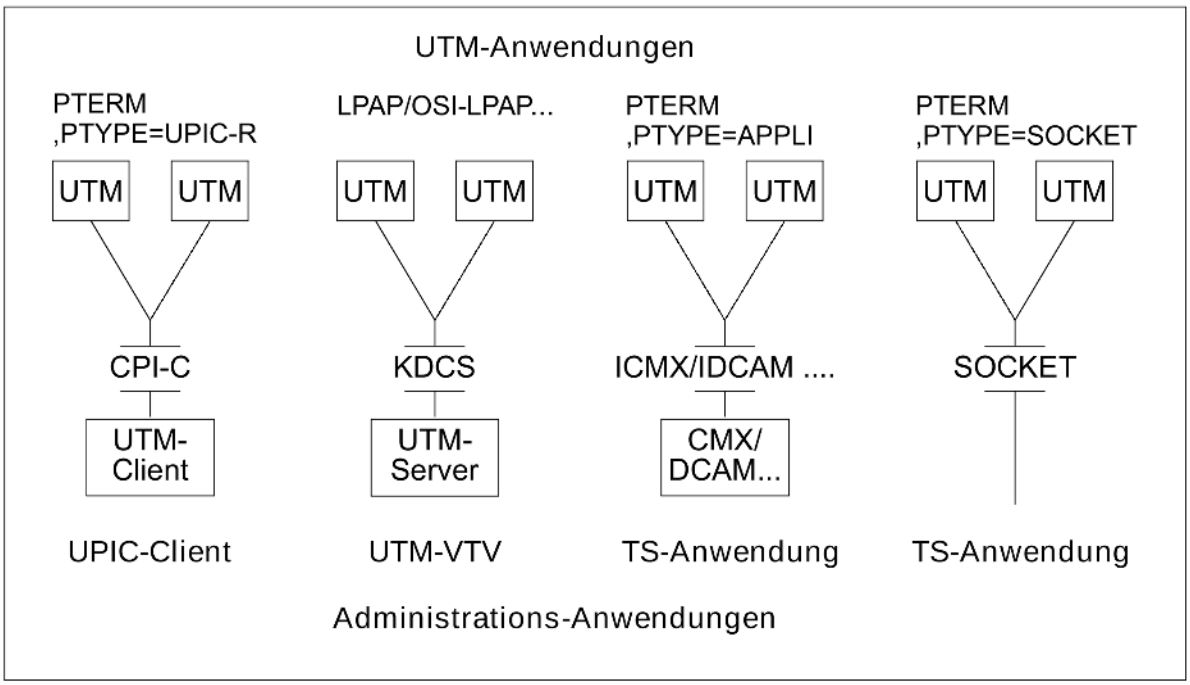
Verfügbarkeit prüfen

Wenn Sie die notwendigen Konfigurationsschritte in UTM und WinAdmin/WebAdmin durchgeführt haben, können Sie die Erreichbarkeit der UTM-Anwendung prüfen.

Ist die Anwendung verfügbar, können Sie sich die Objekte der Anwendung anschauen. Sie werden an der WinAdmin-/WebAdmin-Oberfläche grafisch in einer Baumstruktur bzw. als Tabelle dargestellt.

8.2 Konfigurationsmodelle für eigene Administrationsanwendungen

Die Administrations-Anwendung können Sie als UPIC-Client-Anwendung, als UTM-Anwendung mit verteilter Verarbeitung (mit oder ohne globale Transaktionssicherung) oder als TS-Anwendung (SOCKET, CMX, DCAM, UTM, HTTP-Client) realisieren. Das folgende Bild veranschaulicht alle Möglichkeiten und die verwendeten Schnittstellen.



In allen Fällen muss die Administrations-Anwendung in den administrierten Anwendungen so generiert werden, dass sie die Administrationsberechtigung besitzt.

Das Diagramm gilt sinngemäß auch für die Administration von UTM-Cluster-Anwendungen, siehe auch „Generierungsbeispiel für eine UTM-Cluster-Anwendung“ im Abschnitt "Administration über UPIC-Client".

8.2.1 Administration über UPIC-Client

Ein UPIC-Client kann auf BS2000-, Unix-, Linux- und Windows-Systemen ablaufen. Wenn Sie ein Windows-System wählen, haben Sie den Vorteil, dass Sie für die Administration eine komfortable grafische Oberfläche erstellen können.

Ein Client ist auch wiederanlauffähig, indem er die letzte Ausgabe-Nachricht anfordern und den unterbrochenen Vorgang fortsetzen kann, siehe Handbuch „openUTM-Client für das Trägersystem UPIC“.

Beachten Sie bitte, dass ein UPIC-Client

- zu einer Zeit nur mit einer Anwendung kommunizieren kann, wenn er unter einem BS2000-System abläuft,
- selbst keine Asynchron-Aufträge an openUTM stellen kann,
- immer selbst die Initiative ergreifen muss, d.h. nicht von der administrierten Anwendung aus gestartet werden kann.

> **i** Mit den Produkten WinAdmin und WebAdmin werden UPIC-Clients auf Unix-, Linux- und Windows-Systemen angeboten.
> WinAdmin und WebAdmin bieten den kompletten Funktionsumfang der Programmschnittstelle KDCADMI (siehe Abschnitt „Administration über WinAdmin und WebAdmin").
>
> Mit UTM auf BS2000-Systemen wird ein UPIC-Client mit SDF-Kommandooberfläche in Form eines übersetzten Objektcodes ausgeliefert. Die Konfiguration für dieses Programm können Sie an Ihre Konfiguration anpassen, näheres finden Sie im Abschnitt „CALLUTM - Tool für Administration und Client-Server-Kommunikation (BS2000-Systeme)" im Anhang.

Programmierung

Sie erstellen ein UPIC-Programm, das die für die Administration benötigten Daten (Administrationskommando oder Input für Administrationsprogramm) an die ferne Anwendung sendet und die Ausgabe der administrierten Anwendung empfängt. Das folgende Diagramm skizziert ein UPIC-Programm für Unix-, Linux- oder Windows-Systeme.

```
#include <upic.h>
Enable_UTM_UPIC                       /* Anmelden beim Trägersystem UPIC */
Initialize_Conversation               /* Conversation initialisieren,    */
                                      /* sym_dest_name adressiert die zu */
                                      /* administrierende Anwendung      */
Set_TP_Name                           /* TAC des Administrationsprogramms*/
                                      /* oder Administrationstac KDC.... */

Set_Conversation_Security_Type=CM_SECURITY_PROGRAM
                                      /* UTM-Benutzerkonzept verwenden   */
Set_Conversation_Security_User_ID     /* UTM-Benutzerkennung setzen, die */
                                      /* Admin-Berechtigung haben muss   */
Set_Conversation_Security_Password    /* Passwort fuer Benutzerkennung   */
...
Allocate                              /* Conversation einrichten         */

memcpy (buffer,  )                    /* Datenbereich versorgen mit      */
                                      /* Kommando oder Programminput     */

Send_Data                             /* Kommando/Programminput an die   */
                                      /* administrierte Anwendung senden */

Receive                               /* Rueckmeldung von UTM-Anwendung, */
                                      /* das Programm muss diese         */
                                      /* anschliessend auswerten         */

Disable_UTM_UPIC                      /* Abmelden von Trägersystem UPIC  */
```

Auf welche Arten das UPIC-Programm die Daten senden und empfangen kann, ist im Abschnitt „Zentrale Administration über Kommandos" bzw. im Abschnitt „Zentrale Administration über Programme" beschrieben.

Generierungsbeispiel (stand-alone UTM-Anwendung)

Das UPIC-Programm auf dem Unix- oder Linux-System *UNIX0001* soll drei UTM-Anwendungen administrieren. Eine Anwendung läuft auf dem BS2000-System *D123ZE45*, eine andere auf dem Unix- oder Linux-System *D234S012* und die dritte auf dem Windows-System *WSERV01*. Für die UTM-Anwendungen soll der Administrations-TAC KDCSHUT zum Herunterfahren der Anwendung sowie das Administrationsprogramm mit dem TAC TPADMIN aufrufbar sein.

1. Einträge in der upicfile des UPIC-Clients:

```
* Local Name der CPI-C-Anwendung
LNADMIN001 UPIC0001;
* UTM-Anwendung auf einem BS2000-System
HDUTMAW001 APPLIBS2.D123ZE45 TPADMIN;
* UTM-Anwendung auf einem Unix- oder Linux-System
SDUTMAW002 APPLUnix.D234S012 TPADMIN PORT=30000;
* UTM-Anwendung auf einem Windows-System
SDUTMAW003 APPLIWIN.WSERV01 TPADMIN PORT=30000;
```

2. UTM-Generierung auf dem BS2000-System:

```
BCAMAPPL APPLIBS2,T-PROT=ISO
PTERM UPIC0001,PTYPE=UPIC-R,LTERM=UPICLTRM,
      ,BCAMAPPL=APPLIBS2,PRONAM=UNIX0001,...
LTERM UPICLTRM,KSET=ALLKEYS,USER=REMADMIN,RESTART=N
USER REMADMIN,PERMIT=ADMIN,RESTART=NO                  *)
TAC KDCSHUT, PROGRAM=KDCADM,ADMIN=Y                   **)
TAC TPADMIN,PROGRAM=ADMINPRG,ADMIN=Y,...
PROGRAM ADMINPRG,...
PROGRAM KDCADM
```

   Der Prozessorname *UNIX0001* muss in BCAM generiert werden (per BCIN- bzw. CREATE-PROCESSOR-Kommando oder in der RDF). BCMAP-Einträge sind bei RFC1006 über Port 102 nicht erforderlich.

3. UTM-Generierung auf dem Unix- oder Linux-System:

```
BCAMAPPL APPLUnix,LISTENER-PORT=30000,TSEL-FORMAT=T,T-PROT=RFC1006
PTERM UPIC0001,PRONAM=UNIX0001,TSEL-FORMAT=T,PTYPE=UPIC-R
      ,LTERM=UPICLTRM,BCAMAPPL=APPLUnix
LTERM UPICLTRM,KSET=ALLKEYS,USER=REMADMIN,RESTART=N
USER REMADMIN,PERMIT=ADMIN,RESTART=NO                  *)
TAC KDCSHUT, PROGRAM=KDCADM,ADMIN=Y                   **)
TAC TPADMIN,PROGRAM=ADMINPRG,ADMIN=Y,...
PROGRAM ADMINPRG,...
PROGRAM KDCADM
```

4. UTM-Generierung auf einem Windows-System:

```
BCAMAPPL APPLIWIN,LISTENER-PORT=30000,TSEL-FORMAT=T,T-PROT=RFC1006
PTERM UPIC0001,PRONAM=UNIX0001,TSEL-FORMAT=T
      ,PTYPE=UPIC-R,LTERM=UPICLTRM,BCAMAPPL=APPLIWIN
LTERM UPICLTRM,KSET=ALLKEYS,USER=REMADMIN,RESTART=N
USER REMADMIN,PERMIT=ADMIN,RESTART=NO                    *)
TAC KDCSHUT, PROGRAM=KDCADM,ADMIN=Y                     **)
TAC TPADMIN,PROGRAM=ADMINPRG,ADMIN=Y,...
PROGRAM ADMINPRG,...
PROGRAM KDCADM
```

*) Hier wird mit der Verbindungs-Benutzerkennung gearbeitet, bei der kein Passwort geprüft wird. Wenn man höhere Sicherheit haben möchte, dann muss der UPIC-Client mit den CPI-C-Aufrufen *Set_Conversation_Security_Type/_User_ID/_Password* eine „echte“ Benutzerkennung an openUTM übergeben. Diese muss dann mit Administrationsrechten ausgestattet und durch ein Passwort geschützt werden.

**) Sie sollten alle relevanten Administrations-TACs generieren, KDCSHUT muss auf jeden Fall generiert werden. Im UPIC-Programm kann der TAC per Programm gesetzt werden (Standard ist TPADMIN).

Generierungsbeispiel für eine UTM-Cluster-Anwendung (Unix-, Linux- und Windows-Systeme)

Das UPIC-Programm auf dem Unix- oder Linux-System *UNIX0002* soll eine UTM-Cluster-Anwendung auf den Linux-Systemen *C123DE10*, *C123DE11* und *C123DE12* administrieren, Die UTM-Cluster-Anwendung *APPLLINC* besteht aus drei Knoten und soll durch das Administrationsprogramm mit dem TAC REMADMIN aufrufbar sein.

1. Einträge in der upicfile des UPIC-Clients:

   Der UPIC-Client wird so konfiguriert, dass Sie für jeden Knoten einen eigenen Symbolic Destination Name angeben.

```
* Local Name der CPI-C-Anwendung
LNADMIN001 UPIC0001;
* UTM-Cluster-Anwendung auf dem Linux-System
CDclnode01 APPLLINC.C123DE10  REMADMIN
CDclnode02 APPLLINC.C123DE11  REMADMIN
CDclnode03 APPLLINC.C123DE12  REMADMIN
```

   Das UPIC-Programm muss in diesem Fall den betreffenden Knoten (*clnode01, clnode02* oder *clnode03*) explizit adressieren.

2. UTM-Generierung auf dem Linux-System (initiale KDCFILE):

```
BCAMAPPL APPLLINC,T-PROT=ISO
PTERM UPIC0001,PTYPE=UPIC-R,LTERM=UPICLTRM,
      ,BCAMAPPL=APPLLINC,PRONAM=UNIX0002,...
LTERM UPICLTRM,KSET=ALLKEYS,USER=REMADMIN,RESTART=N

USER ADMUSR01,PERMIT=ADMIN,RESTART=NO                          *)
USER ADMUSR02,PERMIT=ADMIN,RESTART=NO                          *)
USER ADMUSR03,PERMIT=ADMIN,RESTART=NO                          *)
TAC KDCSHUT, PROGRAM=KDCADM,ADMIN=Y                           **)
TAC REMADMIN,PROGRAM=ADMINPRG,ADMIN=Y,...
PROGRAM ADMINPRG,...
PROGRAM KDCADM
```

*) Sie sollten pro Knoten eine Benutzerkennung mit Administrationsrechten generieren, da ein Benutzer in einer UTM-Cluster-Anwendung standardmäßig angemeldet bleibt, wenn die Conversation beendet wird. Das UPIC-Programm muss die Benutzerkennung zuweisen.

**) Sie sollten alle relevanten Administrations-TACs generieren. Im UPIC-Programm kann der TAC per Programm gesetzt werden (Standard ist REMADMIN).

8.2.2 Administration über Verteilte Verarbeitung

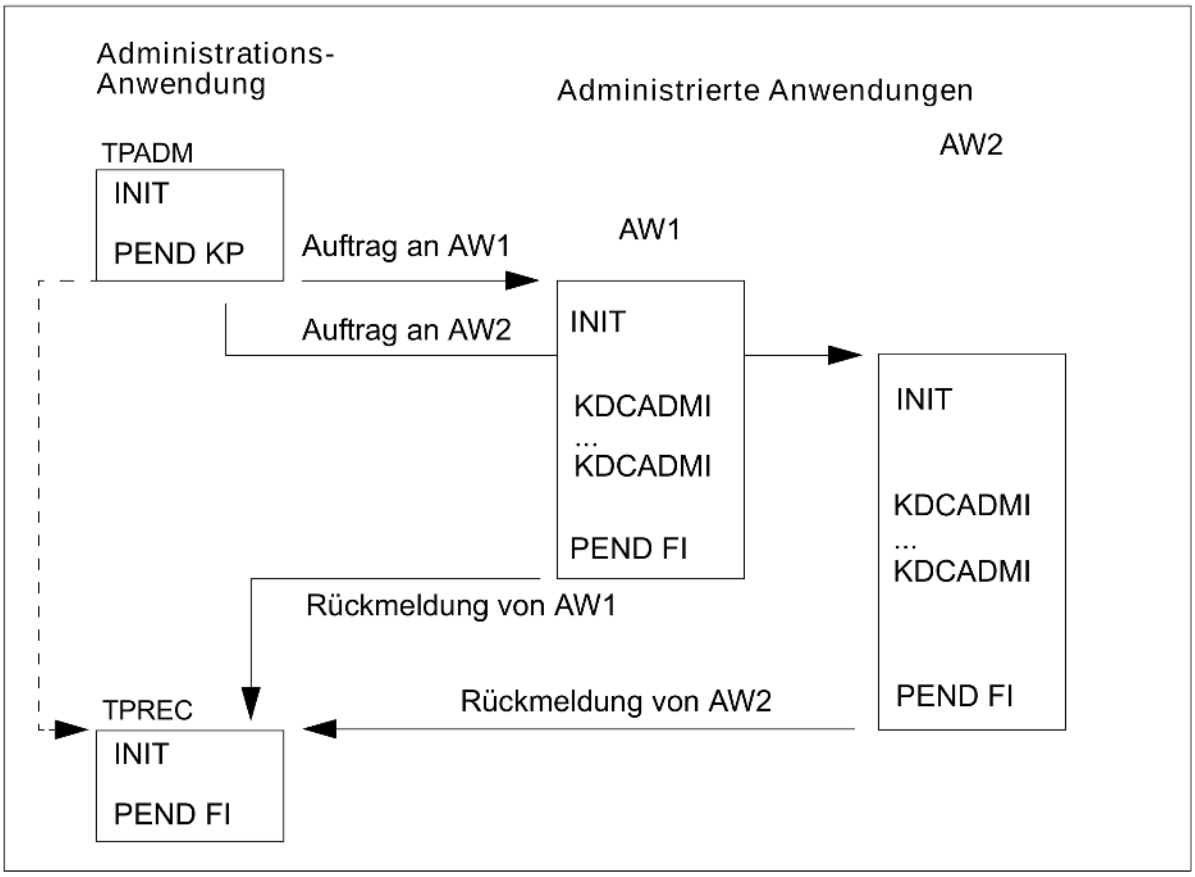
Wenn Sie die zentrale Administration über openUTM mit verteilter Verarbeitung abwickeln möchten, haben Sie den Vorteil, dass

- gleichzeitig mehrere Anwendungen administriert werden können,
- ein Administrations-Auftrag sowohl von der Administrations-Anwendung als auch durch die administrierten Anwendungen angestoßen werden kann (Poll-Funktion),
- auf einfache Art zeitgesteuerte Administrationsaufträge gestellt werden können (DPUT),
- bei Bedarf mit globaler Transaktionssicherung gearbeitet werden kann. Dadurch können Sie z.B. sicherstellen, dass bestimmte Anwendungsparameter gleichzeitig für alle Anwendungen geändert werden, was bei der Administration über einen UPIC-Client oder eine TS-Anwendung nicht gewährleistet ist, da z.B. Netzstörungen dazu führen können, dass die Aktion in einer der Anwendungen nicht durchgeführt werden kann, während die anderen Anwendungen schon mit den neuen Werten arbeiten.

Für die Kommunikation zwischen Administrations-Anwendung und den administrierten Servern können Sie die Protokolle LU6.1 oder OSI TP verwenden.

Programmierung

Wenn Sie für die Administration eine globale Transaktionssicherung wünschen, muss eine Transaktion der Administrations-Anwendung mit mehreren Auftragnehmern kommunizieren. Das folgende Diagramm veranschaulicht das Prinzip anhand von zwei administrierten Anwendungen, die jeweils mehrere Administrationsaufrufe absetzen.



Das Programm TPADM beauftragt beide Anwendungen. Das Programm TPREC wird erst aufgerufen, nachdem die Antworten von beiden Anwendungen eingetroffen sind. Wenn beide Anwendungen ihren Auftrag ordnungsgemäß abgeschlossen haben, beendet TPREC die globale Transaktion und den Vorgang.

Das folgende Beispiel skizziert, wie die Programme TPADM und TPREC aussehen können. Die Administrationsaufgabe besteht darin, von einem Unix- oder Linux-System aus einen Programmaustausch in einer UTM-Anwendung auf einem Unix- oder Linux-System und einer UTM-Anwendung auf einem BS2000-System gleichzeitig zu initiieren. Der Austausch verläuft auf einem Unix- oder Linux-System und auf einem BS2000-System unterschiedlich. Auf einem BS2000-System wird die aktuelle Version des Lademoduls ermittelt, das Lademodul zum Austausch vorgemerkt und anschließend die Anwendung neu geladen. Auf einem Unix- oder Linux-System wird das Programm gleich ausgetauscht. Die administrierten Anwendungen können dabei ein Programm wie im Abschnitt "Mehrere Administrationsaufrufe" verwenden. Das folgende Diagramm skizziert die Umsetzung des Beispiels für LU6.1 und für OSI TP ohne globale Transaktionssicherung.

Programmierung und Generierung sind analog, wenn statt der administrierenden und/oder der administrierten UTM-Anwendung auf Unix- und Linux-Systemen eine UTM-Anwendung auf Windows-Systemen eingesetzt wird. Bitte beachten Sie, dass Portnummer 102 nicht für UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen verwendet werden kann.

```
/* Teilprogramm TPADM sendet Daten an Anwendung UTMAPPL1 und UTMAPPL2     */

INIT
memcpy (buffer,  ...)                  /* Daten vorbereiten               */
APRO DM KCPI=VGID1 KCPA=UTMAPPL1       /* Auftragnehmerservice TPADMIN    */
        KCRN=TPADMIN                   /* in UTMAPPL1 adressieren         */

MPUT NE buffer                         /* Daten an UTMAPPL1 senden        */
     KCRN=VGID1

APRO DM KCPI=VGID2 KCPA=UTMAPPL2       /* Auftragnehmerservice TPADMIN in */
        KCRN=TPADMIN                   /* UTMAPPL2 adressieren            */

MPUT NE buffer                         /* Daten an UTMAPPL2 senden        */
     KCRN=VGID2

PEND KP KCRN=TPREC                     /* Warten auf Auftragnehmer        */
```

Für OSI TP mit globaler Transaktionssicherung sind zusätzliche Anweisungen erforderlich, um

- die Commit Functional Unit auszuwählen
  (APRO... KCOF=C)
- UTMAPPL1 aufzufordern, das Transaktions- und Dialog-Ende einzuleiten
  (CTRL PE, KCRN=VGID1)
- UTMAPPL2 aufzufordern, das Transaktions- und Dialog-Ende einzuleiten
  (CTRL PE, KCRN=VGID1)

```
/* Folgeprogramm TPREC nimmt Quittung von Auftragnehmer-Service entgegen  */

INIT
   KCRPI=VGIDx                          /* 1. Nachricht kommt von AN-      */
                                        /* Service mit Vorgangs-ID VGIDx   */
MGET NT KCRN=VGIDx                      /* Antwort von AN-Service 1 lesen, */
        KCRCCC=12Z KCRPI=VGIDy          /* weitere Nachricht von anderem   */
                                        /* AN-Service (VGIDy) liegt vor    */

if (OK)                                 /* AN-Service1 hat den Programm-   */
 {                                      /* austausch angestossen           */
   MGET NT KCRN=VGIDy                   /* Antwort von AN-Service 2 lesen  */
           KCRCCC=10Z KCRPI=SPACES      /* Keine Nachricht mehr vorhanden  */

  if (OK)                               /* AN-Service2 hat den Programm-   */
    {                                   /* austausch angestossen           */
      MPUT NE                           /* Nachricht an Administrator      */
      PEND FI                           /* Globale Transaktion beenden     */
    } else error_routine();
 } else error_routine();
....
error_routine ()                        /* Fehlerroutine                   */
        { MPUT NE                       /* Administrator benachrichtigen   */
          PEND FR }                     /* Globale Transaktion             */
                                        /* zuruecksetzen und beenden       */
```

Generierungsbeispiel

Das Beispiel zeigt eine LU6.1-Generierung, wobei die Administrations-Anwendung eine zweistufige Adressierung verwendet.

Im Beispiel werden die Portnummern und Rechnernamen (BS20HOST, UnixHOST, UnixADMI) in den Generierungsanweisungen angegeben. Siehe dazu openUTM-Handbuch „Anwendungen generieren“ unter „Adressinformationen bereitstellen“.

1. Generierung der UTM-Administrations-Anwendung auf Unix- oder Linux-Systemen

```
BCAMAPPL ADMINAPP,LISTENER-PORT=1234,T-PROT=RFC1006,T-SEL-FORMAT=T
***
*** Verbindung zur Anwendung auf Unix- oder Linux-System,
*** Administrator-Anwendung ist Auftraggeber
SESCHA ADMAPPL1,PLU=Y,CONNECT=Y
LPAP UTMAPPL1,SESCHA=ADMAPPL1
LSES ADMAG1,LPAP=UTMAPPL1,...
CON APPLUnix,BCAMAPPL=ADMINAPP,PRONAM=UnixHOST -
              ,LISTENER-PORT=2345,LPAP=UTMAPPL1,...
***
*** Verbindung zur Anwendung auf dem BS2000-System,
*** Administrator-Anwendung ist Auftraggeber
SESCHA ADMAPPL2,PLU=Y,CONNECT=Y
LPAP UTMAPPL2,SESCHA=ADMAPPL2
LSES ADMAG2,LPAP=UTMAPPL2,...
CON APPLIBS2,BCAMAPPL=ADMINAPP,PRONAM=BS20HOST -
              ,LISTENER-PORT=102,LPAP=UTMAPPL2,...
***
*** LTAC fuer fernes Administrationsprogramm, zweistufige Adressierung
*** LTAC=RTAC ist der TAC in der entfernten Anwendung
LTAC TPADMIN
***
*** TACs fuer die beiden Administrationsprogramme
TAC TPADM,PROGRAM=...
TAC TPREC,PROGRAM=...
```

2. Generierung der administrierten UTM-Anwendung auf BS2000-Systemen

```
BCAMAPPL APPLIBS2,T-PROT=ISO
***
*** LU6-Generierung fuer Auftragnehmer
SESCHA ADMINREC,PLU=N,CONNECT=N
LPAP UTMADMIN,SESCHA=ADMINREC,PERMIT=ADMIN
LSES ADMAN,LPAP=UTMADMIN,...
CON ADMINAPP,BCAMAPPL=APPLIBS2,PRONAM=UnixADMI,LPAP=UTMADMIN,...
***
TAC TPADMIN,PROGRAM=ADMINPRG,ADMIN=Y
PROGRAM ADMINPRG,...
```

3. Generierung der administrierten UTM-Anwendung auf Unix- und Linux-Systemen

```
BCAMAPPL APPLUnix,LISTENER-PORT=1234,T-PROT=RFC1006,T-SEL-FORMAT=T
***
*** LU6-Generierung fuer Auftragnehmer
SESCHA ADMINREC,PLU=N,CONNECT=N
LPAP UTMADMIN,SESCHA=ADMINREC,PERMIT=ADMIN
LSES ADMAN,LPAP=UTMADMIN,...
CON ADMINAPP,BCAMAPPL=APPLUnix,PRONAM=UnixADMI -
              ,LISTENER-PORT=2345,LPAP=UTMADMIN,...
***
TAC TPADMIN,PROGRAM=ADMINPRG,ADMIN=Y
PROGRAM ADMINPRG,...
```

8.2.3 Administration über TS-Anwendung

Die Anwendung kann eine beliebige TS-Anwendung wie z.B. eine CMX-Anwendung (PTYPE=APPLI) oder eine Socket-USP-Anwendung (PTYPE=SOCKET) sein. Sie können aber auch eine UTM-Anwendung verwenden, die Sie als TS-Anwendung generieren. Die Administrations-Anwendung wird über eine LTERM/PTERM- oder TPOOL-Anweisung an die administrierten UTM-Anwendungen angebunden.

In allen Fällen kann die Anwendung

- gleichzeitig mehrere UTM-Anwendungen administrieren,
- durch die administrierten Anwendungen angestoßen werden.

Wie die Anwendung programmiert werden kann, hängt von der Art der TS-Anwendung ab. Im Falle einer UTM-Anwendung können Sie auch mit DPUT zeitgesteuerte Aufträge an die administrierten Anwendungen stellen.

Um über eine TS-Anwendung zu administrieren, muss entweder

- die Verbindungs-Benutzerkennung administrationsberechtigt sein, z.B.

  ```
  LTERM ADMINLTM,KSET=ALLKEYS,RESTART=N, USER=ADMINUS
  USER ADMINUS, PERMIT=ADMIN, RESTART=N
  ```

  oder
- im Anmelde-Vorgang für die TS-Anwendung muss eine echte Benutzerkennung angemeldet werden, die administrationsberechtigt ist.

Generierung

Für die Generierung einer administrierten UTM-Anwendung auf einem BS2000-System sollen beispielsweise der Aufruf des Kommandos KDCSHUT und des Administrationsprogramms mit dem TAC TPADMIN erlaubt sein.

Dazu sind in der dezentralen Anwendung folgende Anweisungen für LTERM, TAC und PROGRAM notwendig, unabhängig davon, ob die zentrale Anwendung eine Socket, CMX- oder eine DCAM-Anwendung ist:

```
*****************************************************************
*** LTERM, TAC und PROGRAM
*****************************************************************
LTERM ADMINLTM,KSET=ALLKEYS,RESTART=N
USER ADMINLTM, PERMIT=ADMIN,RESTART=N
TAC KDCSHUT, PROGRAM=KDCADM,ADMIN=Y
TAC TPADMIN,PROGRAM=ADMINPRG,ADMIN=Y,...
PROGRAM ADMINPRG,...
PROGRAM KDCADM
```

Für die Adressierung der zentralen Anwendung schreiben Sie folgende Anweisungen, je nachdem, um welche Art Anwendung (DCAM, CMX) es sich handelt. Ist die zentrale Anwendung eine UTM-Anwendung, dann gilt entsprechendes, je nachdem, ob die Anwendung über NEA oder über TCP/IP angebunden ist.

```
*********************************************************************
*** DCAM-Anwendung, die ueber NEA-Protokolle mit
*** openUTM-Anwendungen auf BS2000-Systemen kommuniziert:
*********************************************************************
BCAMAPPL APPLIBS2,T-PROT=NEA
PTERM dcam-name,PTYPE=APPLI,LTERM=ADMINLTM,
BCAMAPPL=APPLIBS2,PRONAM=dcam-rechner
*********************************************************************
*** CMX-Anwendung auf dem Unix- und Linux-System ueber TCP/IP-RFC1006
*********************************************************************
BCAMAPPL APPLUnix,T-PROT=RFC1006
PTERM t-selektor,PTYPE=APPLI,LTERM=ADMINLTM,BCAMAPPL=APPLUnix,
        LISTENER-PORT=portnummer,PRONAM=unix-rechner
*********************************************************************
*** Socket-Anwendung auf dem Unix- und Linux-System
*********************************************************************
BCAMAPPL SOCKETBS,LISTENER-PORT=12000,T-PROT=SOCKET
PTERM SOCKPTRM,PTYPE=SOCKET,LTERM=ADMINLTM, BCAMAPPL=SOCKETBS,
         LISTENER-PORT=portnummer,PRONAM=unix-rechner
```

dcam-name und *dcam-rechner* sind die Namen der DCAM-Anwendung und des Rechners, auf dem die DCAM-Anwendung läuft. *t-selektor* ist der T-Selektor der fernen CMX-Anwendung. *unix-rechner* ist der Name des Rechners, an dem die CMX- bzw. Socket-Anwendung abläuft. *portnummer* ist die Portnummer, an dem die zentrale CMX- bzw. Socket-Anwendung auf Verbindungsaufbauwünsche wartet.

8.3 Zentrale Administration über Kommandos

Neben der Programmschnittstelle bietet openUTM auch die Kommandoschnittstelle zur Administration an. Allerdings stellt Ihnen die Kommandoschnittstelle nur einen Teil der Funktionalität der Programmschnittstelle zur Verfügung.

Für die zentrale Administration können Sie synchrone und asynchrone Kommandos einsetzen. In beiden Fällen muss das zentrale Administrationsprogramm

- das Kommando in der vorgeschriebenen Syntax bereitstellen
- und als Nachricht an die administrierte UTM-Anwendung senden.

Die administrierte Anwendung führt das Kommando so aus, als wenn es innerhalb der eigenen Anwendung gegeben worden wäre. Um die Kommandoausgabe in der zentralen Anwendung auswerten zu können, müssen Sie jedoch die Unterschiede zwischen der synchronen und asynchronen Methode beachten.

Synchrone Kommandos

Wenn Sie für die zentrale Administration synchrone Administrationskommandos verwenden, dann wird die Ausgabe des Kommandos automatisch an den Absender, d.h. an das Administrationsprogramm zurückgeschickt.

Daher ist für die zentrale Administration über synchrone Kommandos jedes Konfigurationsmodell geeignet. Wenn Sie einen UPIC-Client für Windows-Systeme verwenden, können Sie z.B. ein Programm mit Microsoft Visual Studio erstellen, mit dessen Hilfe Sie die Administrationskommandos über eine komfortable Windows-Oberfläche eingeben. Die Antwort von openUTM kann durch das Programm vor der Ausgabe gefiltert werden, so dass Sie nur die für Sie wichtigen Parameter zu sehen bekommen. Die Nachrichtenschnittstelle zu openUTM realisieren Sie dann über ein CPI-C-Programm, wie es im Abschnitt „Administration über UPIC-Client" beschrieben ist.

Asynchrone Kommandos

Wenn Sie für die zentrale Administration asynchrone Administrationskommandos verwenden, dann wird die Ausgabe nicht automatisch an den Absender zurückgeschickt. Daher muss in den dezentralen Anwendungen mit MAX DESTADM das Ziel für die Kommandoausgabe generiert werden.

Ist die zentrale Anwendung eine TS-Anwendung, dann geben Sie in MAX DESTADM den LTERM-Namen der zentralen Anwendung an. Bitte beachten Sie jedoch, dass die zentrale Anwendung diese Ausgabe asynchron erhält, d.h. sie muss den Absender ermitteln.
Wenn Sie mit verteilter Verarbeitung administrieren möchten, müssen Sie mit MAX DESTADM=TAC ein weiteres dezentrales Asynchron-Programm dazwischenschalten, das die Ausgabe empfängt und mit FPUT an die Administrations-Anwendung weiterleitet.

8.4 Zentrale Administration über Programme

Wenn Sie die Programmschnittstelle einsetzen, können Sie die Aufgaben zwischen der Administrations-Anwendung und den administrierten Anwendungen auf zwei Arten aufteilen:

- Dezentrale Administrationsprogramme:
  Sie verwenden die Programmschnittstelle so, dass in der administrierten Anwendung ein vollständiges Administrationsprogramm existiert, das die notwendigen Parameter selbst ermittelt und die Rückgaben selbst auswertet.
- Zentrale Administrationsprogramme:
  Sie verwenden die Programmschnittstelle in der administrierten Anwendung als reine Nachrichtenschnittstelle, d. h. diese bekommt alle Parameter von der Administrations-Anwendung geliefert und sendet das Ergebnis des Aufrufs (Returncodes, Daten) ohne Prüfung zurück.

8.4.1 Dezentrale Administrationsprogramme

Wenn die administrierten Anwendungen vollständige Administrationsprogramme verwenden, wie sie im Kapitel „Erstellen eigener Administrationsprogramme" beschrieben sind, liegt die Steuerung eines Administrations-Vorgangs im Wesentlichen bei der administrierten Anwendung. Das Administrationsprogramm muss damit

- eine empfangene Nachricht interpretieren, die von der Administrations-Anwendung oder - z.B. bei automatischer Administration - von einem Anwendungs-eigenen MSGTAC-Programm kommt
- sämtliche Bereiche für den Administrationsaufruf richtig versorgen
- die Returncodes auswerten und darauf reagieren, d.h. bei Fehlern die Administrations-Anwendung benachrichtigen und eventuell die Transaktion zurücksetzen
- die zurückgegebenen Daten auswerten und entscheiden, welche Daten an die Administrations-Anwendung geschickt werden.

Dabei ist es zu empfehlen, für die verschiedenen Administrationsaufgaben entweder einzelne Teilprogramme zu erstellen oder, falls Sie ein komplettes Administrationsprogramm verwenden, das Programm je nach Aufgabe mit unterschiedlichen TACs anzusprechen. Dann wird die Aufgabe anhand des TAC und nicht anhand der Nachricht ausgewählt.

Portable Administrationsprogramme

Wenn Sie Ihre Administrationsprogramme in verschiedenen Anwendungen einsetzen wollen, die auf verschiedenen Plattformen ablaufen, dann können Sie die Programme so schreiben, dass sie sowohl auf Unix-, Linux- und Windows-Systemen als auch auf BS2000-Systemen ablaufen.

Diese Aufgabe wird Ihnen dadurch erleichtert, dass die Programmschnittstelle auf allen Plattformen die gleichen Datenstrukturen besitzt. Sie müssen jedoch folgende Plattform-spezifische Unterschiede beachten:

- Es gibt einzelne Felder und Unterstrukturen, die nur auf einer Plattform eine Bedeutung besitzen.
  - Beim Auslesen von Daten werden Felder, die für die Plattform nicht relevant sind, immer mit binär null belegt.
  - Beim Modifizieren oder Erzeugen von Objekten müssen die Felder, die für die jeweilige Plattform nicht relevant sind, mit binär null belegt werden. Daher sollte das Programm zuerst die Plattform ermitteln, auf dem es abläuft. Dazu muss es das Feld *system_type* in der Struktur *kc_system_par_str* auswerten, nachdem es KDCADMI mit folgenden Parametern aufgerufen hat:

    opcode=KC_GET_OBJECT
    subcode1=KC_APPLICATION_PAR
    obj_type=KC_SYSTEM_PAR

    Nach dem Ermitteln der Plattform muss das Programm für die eigentlichen Administrationsaufrufe erst die Felder besetzen, die für alle Betriebssysteme gültig sind. Anschließend belegt es die für die jeweilige Plattform notwendigen Felder.
- Die Sortierreihenfolge der Zeichen ist auf BS2000-Systemen und auf Unix-, Linux- und Windows-Systemen unterschiedlich, da BS2000-Systeme in der Regel einen EBCDIC- und Unix-, Linux- sowie Windows-Systeme einen ISO-Code verwenden.
- Auf BS2000-Anwendungen werden für Namen nur Großbuchstaben verwendet, auf Unix-, Linux- und Windows-Systemen dagegen sowohl Klein- als auch Großbuchstaben.
- Unix, Linux- und Windows-Systeme verwenden normalerweise andere Zeichen-Codierungen als BS2000-Systeme (ASCII/EBCDIC Problematik).

Das folgende Beispiel zeigt ein portables Administrationsprogramm, das in der dezentralen Anwendung ein Lademodul, Shared Object bzw. DLL austauscht. Das Programm prüft, auf welcher Plattform es abläuft, und benutzt das Ergebnis, um im Programm entsprechend zu verzweigen.

Auf Unix-, Linux- und Windows-Systemen wird nur das Shared Object/die DLL ausgetauscht, während auf BS2000-Systemen geprüft wird, ob das Lademodul in einem Common Memory Pool liegt und damit ein Anwendungsaustausch notwendig ist.

```
#include <kcadminc.h>       /* Include-Datei fuer die Administration     */
INIT
...
MGET                         /* Name/Daten des Teilprogramms einlesen    */

... Eingabe analysieren
KDCADMI opcode=KC_GET_OBJECT /* Betriebssystem abfragen                  */

KDCADMI opcode=KC_GET_OBJECT
                             /* Aktuelle Version des Lademoduls          */
                             /* ermitteln und prüfen, ob es ueberhaupt   */
                             /* austauschbar ist                         */

if (BS2000)                  /* BS2000-Routine                           */
   { KDCADMI opcode=KC_GET_OBJECT
                             /* Lademodus abfragen und ermitteln, ob das */
                             /* Programm zum Austausch vorgemerkt ist    */
     KDCADMI opcode=KC_MODIFY_OBJECT
                             /* Lademodul austauschen bzw. vormerken,    */
                             /* falls er in Common Memory Pool liegt     */
     if (Common Memory Pool)
         KDCADMI opcode=KC_CHANGE_APPLICATION
                             /* Anwendung austauschen                    */
   }                         /* Ende der BS20000-Routine                 */
else                         /* Unix/Linux/Windows-Routine               */
     KDCADMI opcode=KC_MODIFY_OBJECT
                             /* Shared Object/DLL austauschen             */
                             /* Ende der Unix/Linux/Windows-Routine       */

MPUT                         /* Meldung an Initiator                      */
PEND FI
```

Das Programm kann noch ergänzt werden um dynamische Generierung (TAC, PROGRAM,...) wie im Beispiel im Abschnitt "Mehrere Administrationsaufrufe".

8.4.2 Zentrale Administrationsprogramme

Sie können die Programmschnittstelle auf der Seite der administrierten Anwendungen als reine Nachrichtenschnittstelle verwenden. Die Steuerung der Administration liegt dann ganz bei der Administrations-Anwendung. Diese versorgt die zum jeweiligen Administrationsaufruf notwendigen vier Bereiche vollständig und schickt sie mit MPUT NT/NE an die administrierte Anwendung.

Die administrierte Anwendung setzt die gelieferten Daten nur auf die Syntax der Administrationsschnittstelle um und ruft sie anschließend auf. D.h. sie prüft weder die bei MGET gelieferten Daten noch die Returncodes und Rückgaben des Aufrufs. Das folgende Diagramm skizziert ein solches Programm.

```
/*********************************************************************/
/*         Dialog-Programm fuer die administrierte Anwendung         */
/*                                                                   */
/* Das Programm stellt vier Puffer fuer den Empfang bereit:          */
/* parameter_area, identification_area, selection_area, data_area    */
/*********************************************************************/
INIT

MGET NT in parameter_area           /* vollstaendig versorgter Parameter- */
                                    /* bereich fuer die Administrations-  */
                                    /* schnittstelle                      */
MGET NT in identification_area      /* identification_area ist je nach    */
                                    /* opcode des Parameterbereichs       */
                                    /* versorgt                           */
MGET NT in selection_area           /* selection_area ist abhaengig von   */
                                    /* gewuenschter Aktion versorgt, ggf. */
                                    /* nur mit Laenge 0                   */
MGET NE in data_area                /* Daten, falls erforderlich, sonst   */
                                    /* mit Laenge 0 versorgt              */

KDCADMI (&parameter_area,           /* Das Programm ruft die KDCADMI auf, */
         &identification_area,      /* ohne die Daten zu pruefen          */
         &selection_area,
         &data_area);

MPUT NT parameter_area              /* parameter_area mit den Returncodes */
                                    /* und den anderen Rueckgaben         */
MPUT NE data_area                   /* data_area mit den Rueckgabedaten   */
                                    /* oder Laenge 0, falls keine Daten   */
                                    /* zurueckgegeben werden              */

PEND FI                             /* Vorgang beenden, Information geht  */
                                    /* zurueck an Administrationsanwendung*/
```

Die Administrations-Anwendung muss entsprechend viele Nachrichtenteile senden. Bei einem UPIC-Client kann das z.B. so aussehen:

```
/*************************************************************************/
/* UPIC-Programm fuer die Administrations-Anwendung                      */
/*                                                                       */
/* Das Programm sendet vier Nachrichtenteile                             */
/*************************************************************************/
Enable_UTM_UPIC
Initialize_Conversation
[Set_TP_Name]                       /* ggf. TAC setzen                   */
Set_Conversation_Security_Type      /* Administrationsberechtigten User  */
Set_Conversation_Security_User_ID   /* bei openUTM anmelden              */
Set_Conversation_Security_Password

memcpy  (...)                       /* Alle Datenbereiche versorgen      */
...
memcpy  (...)

Send_Data parameter_area            /* Parameterbereich senden           */
Send_Data identification_area       /* Identifikationsbereich senden     */
Send_Data selection_ara             /* Selektionsbereich senden          */
Send_Data data_area                 /* Datenbereich senden               */
Receive parameter_area              /* Enthaelt die Returncodes/Rueckgaben*/
Receive data_area                   /* Datenbereich mit gewuenschter     */
                                    /* Information                       */

Disable_UTM_UPIC
```

Wie ein solcher UPIC-Client generiert werden kann, ist im Abschnitt "Administration über UPIC-Client" beschrieben.

Wenn die Administrations-Anwendung auf einer anderen Plattform als die administrierte Anwendung läuft, werden die Zeichen der gelieferten Bereiche eventuell umcodiert. Solange diese Bereiche nur abdruckbare Zeichen enthalten wie z.B. Identifikationsbereich, Selektionsbereich und Datenbereich, treten keine Probleme auf. Beim Parameterbereich (*parameter_area*), der auch nicht-abdruckbare Zeichen und numerische Werte enthält, müssen Sie einen Umsetz-Mechanismus verwenden:

- In beiden Anwendungen definieren Sie einen eigenen Parameterbereich, der nur abdruckbare Zeichen enthält.
- Die Administrations-Anwendung setzt die Werte des richtigen Parameterbereichs in abdruckbare Zeichen um, legt sie im eigenen Parameterbereich ab und schickt diesen an die administrierte Anwendung.
- Die administrierten Anwendungen schreiben die empfangenen Werte in den eigenen Parameterbereich, setzen sie in die richtigen numerischen Werte um und kopieren sie in den Parameterbereich, der für den Administrationsaufruf verwendet wird.

9 Administration automatisieren

Sie können Asynchron-Programme oder Administrationskommandos dazu benutzen, die Administration der Anwendung zu automatisieren. Dies kann darin bestehen, dass Parameter lastabhängig herauf- oder heruntergesetzt werden oder dass auf Fehler entsprechend reagiert wird. Zur Steuerung können Sie z.B. ein MSGTAC-Programm und/oder zeitgesteuerte Aufträge verwenden.

Die Steuerung über ein MSGTAC-Programm verläuft nach folgendem Schema:

1. In der Anwendung tritt ein Ereignis auf, das eine bestimmte Meldung erzeugt.
2. Diese Meldung wird an das MSGTAC-Programm weitergeleitet.
3. Das MSGTAC-Programm analysiert die Meldung und stößt daraufhin entsprechende Aktionen an.

Mögliche Aktionen sind z.B. der Aufruf der Programmschnittstelle KDCADMI, der Aufruf eines Administrationskommandos oder der Start eines Asynchron-Administrationsprogramms (FPUT/DPUT), das weitere Administrationsaufgaben ausführt.

Anstelle des MSGTAC-Programms kann auch ein Programm verwendet werden, dem ein TAC zugeordnet wird, der als zusätzliches Meldungsziel definiert ist (KDCDEF-Anweisung MSG-DEST).

Wenn Sie WinAdmin oder WebAdmin als Administrationstool einsetzen, dann können Sie dieses auch dazu verwenden, bei bestimmten Ereignissen - z.B. Überschreitung eines Schwellwertes - Skripte auszuführen oder Programme zu starten.

Eine andere Möglichkeit der ereignisgesteuerten Administration besteht darin, in regelmäßigen Abständen bestimmte Statistikdaten abzufragen und entsprechend darauf zu reagieren.

Ein weiterer Einsatzfall ist die Diagnose. Sie können bei bestimmten Ereignissen z.B. den Testmodus einschalten, Traces erstellen, einen UTM-Dump erzeugen oder die Zulieferung an den Messmonitor openSM2 einschalten.

9.1 Steuerung über MSGTAC-Programm

Wie Sie die Administration mit Hilfe eines MSGTAC-Programms automatisieren können, wird anhand eines Beispiels gezeigt. In diesem Beispiel reagiert die Anwendung auf die Meldung `K041 Warnstufe xx fuer Pagepool wurde ueberschritten`. Statt K041 können Sie auch weitere Meldungen wie z.B. `K091... Betriebsmittelengpaß...` zur Steuerung verwenden.

Für dieses Beispiel muss für K041 das Meldungsziel MSGTAC definiert sein und ein MSGTAC-Programm geschrieben und generiert werden. Dieses Programm liest die Meldung und startet im Beispiel mit einer FPUT-Ausgabe-Nachricht das Asynchron-Programm PRGK041.

Im Folgenden wird PRGK041 in zwei Versionen gezeigt: einmal, wie es die Administrationsaktionen über die Programmschnittstelle und einmal, wie es diese über die Kommandoschnittstelle durchführt. Die Funktionalität eines solchen Programms kann auch in der MSGTAC-Routine selber implementiert werden.

Aufbau des MSGTAC-Programms

Das MSGTAC-Programm kann nach folgendem Schema aufgebaut sein:

```
/*************************** MSGTAC-Programm ***************************/
#include <kcmsg.h>

INIT
FGET meldung                      /* Meldung lesen                      */
...
switch (msg-id)
   { case Kxx:...
     case K041:
       { FPUT daten KCRN=PRGK041 /* Teilprogramm PRGK041 aufrufen       */
         break;
      ...
     case Kyy:..
   }
PEND FI
```

Das Programm PRGK041 steuert die Aktionen, die aufgrund der Meldung K041 notwendig sind. Im Folgenden wird skizziert, wie PRGK041 unter Verwendung der Programmschnittstelle und der Kommandoschnittstelle aussehen kann.

Administrationsaktionen über Programmschnittstelle

Mit MSGTAC wird folgendes asynchrone Administrationsprogramm gestartet.

```
/******** Teilproramm PRGK041 fuer KDCADMI-Programmschnittstelle *********/
#include <kcadminc.h>        /* Include-Datei fuer die Administration    */
INIT
FGET daten                   /* Von MSGTAC gelieferte Daten lesen        */
KDCADMI opcode=KC_GET_OBJECT
                             /* Administrationsaufruf, openUTM liefert   */
                             /* die gewünschten Statistikdaten an das    */
                             /* Programm                                 */

 if {...       }             /* Daten auswerten und Aktionen vorbereiten */

KDCADMI opcode=KC_MODIFY_OBJECT
                             /* Geeigneter Parameter wird modifiziert    */
                             /* Ggf. sind zusaetzliche KDCADMI-Aufrufe   */
                             /* noetig, um weitere Parameter zu aendern  */

FPUT                         /* Eventuell Nachricht an Administrator     */

PEND FI
```

Das Lesen und die Auswertung der Anwendungsdaten können Sie innerhalb eines Programms realisieren, wobei beliebig viele KDCADMI-Aufrufe erlaubt sind. Damit können Sie z.B. mehrere Anwendungsparameter ändern, wenn dies aufgrund der aktuellen Anwendungsdaten notwendig sein sollte.

Beispiel: Diagnosehilfen ein/ausschalten

Das folgende Beispiel reagiert auf die Meldung `K119 OSI TP Fehlerinformation...` Ein MSGTAC-Programm wie z.B. das im Abschnitt "Aufbau des MSGTAC-Programms" skizzierte fängt K119 ab und startet mit FPUT das Administrationsprogramm. Dieses schaltet abhängig von der in K119 gelieferten Information die OSI-Tracefunktionen ein.

```
#include <kcadminc.h>        /* Include-Datei fuer die Administration    */
...
INIT

FGET                         /* Daten von MSGTAC lesen                   */
  if {...        }           /* Daten auswerten                          */

KDCADMI opcode=KC_MODIFY_OBJECT
                             /* Unter bestimmten Voraussetzungen die     */
                             /* OSI-Tracefunktionen einschalten          */

FPUT KCRN=admin-lterm        /* Nachricht an Administrator: Trace laeuft */

DPUT KCRN=TRACEOFF           /* Nach einiger Zeit soll ein weiteres      */
                             /* Asynchron-Programm (TRACEOFF) den Trace  */
                             /* wieder ausschalten                       */

PEND FI
```

Diesen Programmaufbau können Sie auch verwenden, um z.B. auf die Meldung `K065 Netzmeldung...` zu reagieren. Nach dem gleichen Muster schreiben Sie ein Programm, das aufgrund einer bestimmten Meldung einen UTM-Dump erzeugt mit KDCADMI *opcode*=KC_CREATE_DUMP.

9.2 Steuerung über benutzerspezifische Meldungsziele

Für die von openUTM erzeugten Meldungen stellt openUTM vier weitere frei verfügbare Meldungsziele bereit, die Sie zur Steuerung von administrativen Aufgaben heranziehen können. Diese Meldungsziele werden mit USER-DEST-1, USER-DEST-2, USER-DEST-3, USER-DEST-4 bezeichnet, denen Sie explizit folgende Objekte zuweisen können:

- eine USER-Queue (die Message Queue einer Benutzerkennung),
- eine TAC-Queue,
- einen Asynchron-TAC oder
- einen LTERM-Partner, der keinem UPIC-Client zugeordnet ist.

Mit Hilfe dieser Meldungsziele können Sie Meldungen wie Nachrichten in einer TAC- oder USER-Queue über die KDCS-Programmschnittstelle mit der Funktion DGET lesen. Mit dieser Funktion und entsprechender Folgeverarbeitung können Sie MSGTAC-ähnliche Programme entwerfen, die spezifisch auf eine Nachricht reagieren.

Durch das Zuweisen einer USER- oder TAC-Queue zu einem Benutzer-spezifischen Meldungsziel können Sie beispielsweise erreichen, dass UTM-Meldungen an den WinAdmin- oder WebAdmin-Administrationsarbeitsplatz ausgegeben werden (siehe openUTM-Handbuch „Meldungen, Test und Diagnose“ oder die Online-Hilfe zu WinAdmin/WebAdmin, Stichwort „Meldungskollektor“).

Die Benutzer-spezifischen Meldungsziele werden mit der Generierungsanweisung MSG-DEST konfiguriert. Spezifische Informationen zu einem Meldungsziel erhalten Sie mit der Anweisung KC_GET_OBJECT Objekttyp KC_MSG_DEST_PAR.

Die Zuordnung einer Meldung zu einem Meldungsziel nehmen Sie mit Hilfe des Dienstprogramms KDCMMOD vor. Welche Meldungen den Benutzer-spezifischen Meldungszielen zugeordnet werden können, ist im openUTM-Handbuch „Meldungen, Test und Diagnose“ beschrieben.

Beim Auftreten einer Meldung, für die als Meldungsziel USER-DEST-*n* definiert ist, erzeugt openUTM einen Asynchron-Auftrag an dieses Meldungsziel.

Wird der Asynchron-Auftrag zurückgewiesen, weil beispielsweise das zugewiesene Objekt gesperrt ist, dann geht die Meldung für das Meldungsziel verloren. Bei einer nächsten Meldung für das Meldungsziel versucht openUTM erneut, einen Asynchron-Auftrag für dieses Meldungsziel zu erzeugen.

Ist einem Meldungsziel USER-DEST-*n* ein Asynchron-TAC zugewiesen, dann startet openUTM das Programm, das dem TAC zugeordnet ist, für jede erzeugte Meldung einmal. Anders als bei MSGTAC, kann in einem Programmlauf immer nur eine Meldung mit FGET gelesen werden. Für diesen Teilprogrammlauf sind im KB-Kopf als Benutzer KDCMSGUS und als LTERM KDCMSGLT eingetragen.

10 Zugriffsrechte und Zugriffsschutz

Die Administrationsberechtigung wird in der UTM-Generierung festgelegt. Sie ist nicht an eine bestimmte Person (Benutzerkennung) oder an einen bestimmten Ort (Konsole) gebunden. Die Administration kann über jeden beliebigen LTERM-Partner erfolgen, egal ob Terminal, UPIC-Client, HTTP-Client oder TS-Anwendung. Sie können die Administrationsberechtigung auch Partner-Anwendungen Ihrer UTM-Anwendung zuordnen. Jede UTM-Anwendung kann somit von einer anderen Anwendung aus administriert werden. Insbesondere können mehrere Anwendungen auf verschiedenen Rechnern zentral von einer Anwendung aus administriert werden (siehe Kapitel „Zentrale Administration mehrerer Anwendungen").

Speziell für die Administration einer UTM-Anwendung über die Programmschnittstelle KDCADMI und über die Administrationskommandos stellt openUTM - zusätzlich zu den allgemeinen Security-Funktionen (Zugang über Benutzerkennungen, Lock-/Keycode- und Access List-Konzept) - ein zweistufiges Berechtigungskonzept zur Verfügung.

Berechtigungsstufe 1

Administrations-Vorgänge können von Benutzern, Clients und Partner-Anwendungen **ohne** Administrationsberechtigung aufgerufen werden, wenn sie nur die angebotenen Informationen über Objekte und Anwendungsparameter abfragen, diese Informationen sammeln und auswerten (also nur **lesend** auf die Konfigurationsdaten zugreifen). Voraussetzung ist, dass Sie den Transaktionscodes, über die diese Administrations-Vorgänge aufgerufen werden, die Berechtigungsstufe ADMIN=READ zuordnen.

Die Angabe von ADMIN=READ ist nur erlaubt:

- bei den Kommandos KDCINF, KDCINFA, KDCHELP und KDCHELPA
- bei Transaktionscodes von Programmläufen, in denen folgende Aufrufe abgesetzt werden:
  - KC_GET_OBJECT
  - KC_ENCRYPT mit *subopcode1*=KC_READ_ACTIV_PUBLIC_KEY oder *subop-code1*=KC_READ_NEW_PUBLIC_KEY
  - KC_SYSLOG mit *subopcode1*=KC_INFO

Teilprogramme und Transaktionscodes können in diesem Fall wie folgt generiert werden:

- BS2000-Systeme

```
PROGRAM ADMPROG,COMP=ILCS
TAC ADMTAC,PROGRAM=ADMPROG,ADMIN=READ
```

- Unix-, Linux- und Windows-Systeme

```
PROGRAM ADMPROG,COMP=C
TAC ADMTAC,PROGRAM=ADMPROG,ADMIN=READ
```

Berechtigungsstufe 2

Administrations-Vorgänge, die die Anwendungsparameter ändern, Objekte modifizieren, neu eintragen oder löschen (also **schreibend** auf die Konfigurationsdaten zugreifen), können grundsätzlich nur von Benutzerkennungen und Partner-Anwendungen **mit** Administrationsberechtigung aufgerufen werden (PERMIT=ADMIN). Die Transaktionscodes dieser Vorgänge müssen mit ADMIN=Y konfiguriert werden.

Teilprogramme und Transaktionscodes müssen dann wie folgt in die Konfiguration eingetragen werden:

- BS2000-Systeme

```
PROGRAM ADMPROG,COMP=ILCS
TAC ADMTAC,PROGRAM=ADMPROG,ADMIN=Y
```

- Unix-, Linux- und Windows-Systeme

```
PROGRAM ADMPROG,COMP=C
TAC ADMTAC,PROGRAM=ADMPROG,ADMIN=Y
```

Mit ADMIN=Y müssen Sie folgende Transaktionscodes generieren:

- alle Administrationskommandos außer KDCINF[A] und KDCHELP[A]
- Transaktionscodes, die Programmläufe starten, in denen auch andere KDCADMI-Aufrufe als KC_GET_OBJECT,
  KC_ENCRYPT mit *subopcode1*=KC_READ_ACTIV_PUBLIC_KEY oder
  subop-code1=KC_READ_NEW_PUBLIC_KEY oder
  KC_SYSLOG mit *subopcode1*=KC_INFO abgesetzt werden.

Alle Teilprogramme, die Transaktionscodes der Berechtigungsstufe 2 aufrufen, müssen unter einer Benutzerkennung ablaufen, die administrationsberechtigt ist.

Beispiel

Sie können ein Administrationsprogramm erstellen, das, wenn es über den Transaktionscode ADMTAC1 aufgerufen wird, nur abfragt, ob ein Drucker mit der Anwendung verbunden ist. Wird dasselbe Programm über den Transaktionscode ADMTAC2 aufgerufen, dann fragt das Teilprogramm ebenfalls mit KC_GET_OBJECT ab, ob der Drucker mit der Anwendung verbunden ist. Ist der Drucker jedoch nicht mit der Anwendung verbunden, dann fordert das Teilprogramm den Verbindungsaufbau zum Drucker an (KC_MODIFY_OBJECT). ADMTAC1 darf über jede Benutzerkennung und von jeder Partner-Anwendung aufgerufen werden. ADMTAC2 darf nur von Benutzerkennungen und Partner-Anwendungen aufgerufen werden, die administrationsberechtigt sind.

Die zugehörige KDCDEF-Generierung sieht dann folgendermaßen aus:

- BS2000-Systeme

```
PROGRAM ADMPROG,COMP=ILCS
TAC ADMTAC1,PROGRAM=ADMPROG,ADMIN=READ
TAC ADMTAC2,PROGRAM=ADMPROG,ADMIN=Y
```

- Unix-, Linux- und Windows-Systeme

```
PROGRAM ADMPROG,COMP=C
TAC ADMTAC1,PROGRAM=ADMPROG,ADMIN=READ
TAC ADMTAC2,PROGRAM=ADMPROG,ADMIN=Y
```

Eine feinere Differenzierung der Zugriffsrechte können Sie dann mit Hilfe des Lock-/Keycode- und Access List-Konzepts realisieren.

10.1 Konfiguration der Administrator-Verbindung

Die Verbindung, über die ein Administrator eine UTM-Anwendung lokal administriert, kann auf unterschiedliche Art generiert werden. Möglich ist eine Generierung der Verbindung über

- eine TPOOL-Anweisung
- eine PTERM- und LTERM-Anweisung

Empfehlung

Die Verbindung des (Haupt-) Administrators sollte über eine PTERM- und LTERM-Anweisung generiert werden. Eine solche Verbindung bietet zum einen eine höhere Sicherheit gegen einen unberechtigten Zugang als ein offener Terminal-Pool. Zum anderen kann ein explizit als Administrator-Arbeitsplatz generiertes LTERM mittels folgender Anweisung als privilegiert gekennzeichnet werden:

```
MAX PRIVILEGED-LTERM = lterm-name
```

Eine so generierte Verbindung wird von UTM in Engpass-Situationen bevorzugt behandelt, so dass einem Administrator der Zugriff auf einer Anwendung, die unter Last arbeitet, erleichtert wird.

10.2 Administrationsberechtigung erteilen

Anwendungen mit Benutzerkennungen

In Anwendungen mit Benutzerkennungen können Transaktionscodes der Berechtigungsstufe 2 nur unter Benutzerkennungen und Partner-Anwendungen aufgerufen werden, denen beim Eintragen in die Konfiguration die Administrationsberechtigung zugeordnet wurde. Die Benutzerkennungen und Partner-Anwendungen, die die lokale Anwendung administrieren sollen, müssen wie folgt generiert werden:

```
USER ADMUS,[PASS=C'.....',PROTECT-PW=(...,....,...)], PERMIT=ADMIN.....
LPAP ADMPA,SESCHA=...,PERMIT=ADMIN....
OSI-LPAP ADMPAO,ASS-NAMES=...,CONTWIN=...,PERMIT=ADMIN....
```

Über eine OSI TP-Partner-Anwendung kann man auch administrieren, wenn das OSI-LPAP keine Administrationsberechtigung hat. In diesem Fall muss im Application-Context des OSI-LPAP die abstrakte Syntax UTMSEC enthalten sein und der Partner eine Benutzerkennung übergeben, die in der lokalen Anwendung administrationsberechtigt ist.

Benutzerkennungen mit Administrationsberechtigung können auch dynamisch in die Konfiguration der Anwendung aufgenommen werden.

Anwendungen ohne Benutzerkennungen

In Anwendungen ohne Benutzerkennungen kann jeder Benutzer oder Client, der sich über einen LTERM-Partner an die Anwendung anschließt, Administrationskommandos und andere Administrations-TACs ausführen. Einen Zugriffsschutz auf diese Vorgänge können Sie dann lediglich über das Lock-/Keycode- und Access List-Konzept realisieren. Sie müssen dazu die Administrationskommandos mit einem Lockcode oder einer Access Liste schützen. Ein Keyset mit einem passenden Keycode vergeben Sie dann nur an Clients und Terminals (LTERM-Partner), über die die Administration der Anwendung möglich sein soll. Partner-Anwendungen dürfen auch in Anwendungen ohne Benutzerkennungen Administrationsfunktionen der Berechtigungsstufe 2 nur ausführen, wenn sie mit PERMIT=ADMIN generiert sind.

10.3 Administrationskommandos generieren

Die Administrationskommandos von openUTM, die Sie im Betrieb der Anwendung nutzen wollen, müssen Sie entweder bei der KDCDEF-Generierung angeben oder dynamisch mit WinAdmin, WebAdmin oder einem selbst erstellten Administrationsprogramm in die Konfiguration eintragen.

Dazu müssen Sie das Administrationsprogramm KDCADM (PROGRAM-Anweisung) definieren und die benötigten Kommandos als Transaktionscodes von KDCADM generieren.

Im Folgenden ist eine vollständige KDCDEF-Generierung von KDCADM und aller Administrationskommandos aufgeführt. Sie müssen in Ihrer KDCDEF-Generierung nur die TAC-Anweisungen der Administrationskommandos aufnehmen, die Sie im Betrieb nutzen wollen. Das Administrationskommando KDCSHUT müssen Sie jedoch immer generieren.

```
REMARK  KDCADM für openUTM auf BS2000-Systemen generieren:
PROGRAM KDCADM,COMP=ILCS
REMARK  KDCADM für openUTM auf Unix-, Linux- und Windows-Systemen generieren:
PROGRAM KDCADM,COMP=C

REMARK Dialog-TACs (Kommandos) von KDCADM generieren:

TAC KDCAPPL ,PROGRAM=KDCADM,ADMIN=Y
TAC KDCBNDL ,PROGRAM=KDCADM,ADMIN=Y
TAC KDCDIAG ,PROGRAM=KDCADM,ADMIN=Y
TAC KDCHELP ,PROGRAM=KDCADM,ADMIN=READ        "Es ist auch ADMIN=Y erlaubt"
TAC KDCINF  ,PROGRAM=KDCADM,ADMIN=READ        "Es ist auch ADMIN=Y erlaubt"
TAC KDCLOG  ,PROGRAM=KDCADM,ADMIN=Y
TAC KDCLPAP ,PROGRAM=KDCADM,ADMIN=Y
TAC KDCLSES ,PROGRAM=KDCADM,ADMIN=Y
TAC KDCLTAC ,PROGRAM=KDCADM,ADMIN=Y
TAC KDCLTERM,PROGRAM=KDCADM,ADMIN=Y
TAC KDCPOOL ,PROGRAM=KDCADM,ADMIN=Y
TAC KDCPROG ,PROGRAM=KDCADM,ADMIN=Y
TAC KDCPTERM,PROGRAM=KDCADM,ADMIN=Y
TAC KDCSHUT ,PROGRAM=KDCADM,ADMIN=Y
TAC KDCSLOG ,PROGRAM=KDCADM,ADMIN=Y
TAC KDCSWTCH,PROGRAM=KDCADM,ADMIN=Y
TAC KDCTAC  ,PROGRAM=KDCADM,ADMIN=Y
TAC KDCTCL  ,PROGRAM=KDCADM,ADMIN=Y
TAC KDCUSER ,PROGRAM=KDCADM,ADMIN=Y

TAC KDCMUX  ,PROGRAM=KDCADM,ADMIN=Y           "Nur auf BS2000-Systemen"
TAC KDCSEND ,PROGRAM=KDCADM,ADMIN=Y           "Nur auf BS2000-Systemen"

REMARK Asynchron-TACs (Kommandos) von KDCADM generieren:

TAC KDCAPPLA,PROGRAM=KDCADM,ADMIN=Y,TYPE=A
TAC KDCBNDLA,PROGRAM=KDCADM,ADMIN=Y,TYPE=A
TAC KDCDIAGA,PROGRAM=KDCADM,ADMIN=Y,TYPE=A
TAC KDCHELPA,PROGRAM=KDCADM,ADMIN=READ,TYPE=A  "Es ist auch ADMIN=Y erlaubt"
TAC KDCINFA ,PROGRAM=KDCADM,ADMIN=READ,TYPE=A  "Es ist auch ADMIN=Y erlaubt"
TAC KDCLOGA ,PROGRAM=KDCADM,ADMIN=Y,TYPE=A
TAC KDCLPAPA,PROGRAM=KDCADM,ADMIN=Y,TYPE=A
TAC KDCLSESA,PROGRAM=KDCADM,ADMIN=Y,TYPE=A
TAC KDCLTACA,PROGRAM=KDCADM,ADMIN=Y,TYPE=A
TAC KDCLTRMA,PROGRAM=KDCADM,ADMIN=Y,TYPE=A
TAC KDCPOOLA,PROGRAM=KDCADM,ADMIN=Y,TYPE=A
```

```
TAC KDCPROGA,PROGRAM=KDCADM,ADMIN=Y,TYPE=A
TAC KDCPTRMA,PROGRAM=KDCADM,ADMIN=Y,TYPE=A
TAC KDCSHUTA,PROGRAM=KDCADM,ADMIN=Y,TYPE=A
TAC KDCSLOGA,PROGRAM=KDCADM,ADMIN=Y,TYPE=A
TAC KDCSWCHA,PROGRAM=KDCADM,ADMIN=Y,TYPE=A
TAC KDCTACA,PROGRAM=KDCADM,ADMIN=Y,TYPE=A
TAC KDCTCLA,PROGRAM=KDCADM,ADMIN=Y,TYPE=A
TAC KDCUSERA,PROGRAM=KDCADM,ADMIN=Y,TYPE=A

TAC KDCMUXA,PROGRAM=KDCADM,ADMIN=Y,TYPE=A     "Nur auf BS2000-Systemen"
TAC KDCSENDA ,PROGRAM=KDCADM,ADMIN=Y,TYPE=A   "Nur auf BS2000-Systemen"
```

Die Kommandos KDCINF[A] und KDCHELP[A] können - so wie oben mit ADMIN=READ generiert - von jeder Benutzerkennung und von jeder Partner-Anwendung aus aufgerufen werden. Sie können diesen Kommandos aber auch einen Lockcode zuordnen (mit dem Operanden LOCK; z.B. LOCK=1). Dann können diese Kommandos nur von Benutzerkennungen und Partner-Anwendungen aufgerufen werden, denen ein Keyset mit dem zugehörigen Keycode (Keycode 1) zugeordnet ist.

Eine andere Möglichkeit, den Zugriff auf diese Kommandos zu steuern, bietet das Access List-Konzept. Einer Access Liste wird ein Keyset zugewiesen, das mehrere Key-/Zugangscodes enthält, die z.B. spezifisch für eine Gruppe von Kommandos sein können. Wird eine solche Access Liste einem Kommando zugeordnet, kann auf dieses Kommando nur ein Benutzer zugreifen, wenn das Keyset seiner Benutzerkennung und das Keyset des LTERM-Partners, über den der Benutzer angemeldet ist, jeweils mindestens einen Key-/Zugangscode enthält, der auch in der Access Liste des Kommandos enthalten ist.

Sie können die Administrationskommandos dynamisch erzeugen, indem Sie die benötigten Kommandos mit KC_CREATE_OBJECT und *obj_type* KC_TAC erzeugen.

11 Programmschnittstelle zur Administration - KDCADMI

In diesem Kapitel wird die Programmschnittstelle zur Administration in C/C++ beschrieben. Die COBOL-Programmschnittstelle entspricht weitgehend der C/C++-Programmschnittstelle. Aus diesem Grund können Sie die folgende Schnittstellenbeschreibung auch bei der Erstellung von Administrationsprogrammen in COBOL zu Rate ziehen. Besonderheiten, die Sie bei der Erstellung von COBOL-Programmen beachten müssen, sind im Anhang im Abschnitt "Programmschnittstelle zur Administration in COBOL" beschrieben.

An der Schnittstelle werden auf allen unterstützten Plattformen dieselben C- bzw. COBOL-Datenstrukturen übergeben. Felder, die in einem Betriebssystem ohne Bedeutung sind, werden mit binär null belegt.

Die C-Datenstrukturen sind auf Unix-, Linux- und Windows-Systemen in der Include-Datei *kcadminc.h* definiert, auf BS2000-Systemen im Include-Element *kcadminc.h* der Bibliothek SYSLIB.UTM.070.C.

In diesem Kapitel finden Sie:

- eine allgemeine Beschreibung eines KDCADMI-Funktionsaufrufs und der Datenbereiche, die Sie beim Aufruf an UTM übergeben müssen.
- für jeden Operationscode von KDCADMI sind die Operationen beschrieben, die Sie ausführen können, und die Parameterwerte, die Sie dazu an UTM übergeben müssen, sowie die von UTM zurückgelieferten Werte.

  Diese Beschreibung ist alphabetisch nach Operationscodes geordnet.
- die Beschreibung der C-Datenstrukturen, in denen Eigenschaften von Anwendungsobjekten und Anwendungsparametern an der Programmschnittstelle übergeben werden. Dieser Teil informiert zunächst über die Datenstrukturen für Anwendungsobjekte und anschließend über die Datenstrukturen für die Anwendungsparameter. Diese Beschreibungen sind alphabetisch nach den Namen der Datenstrukturen geordnet.
- eine ausführliche Beschreibung wie die Wirkung des KDCADMI-Aufrufs in einer stand-alone UTM-Anwendung und in einer UTM-Cluster-Anwendung ist.

11.1 Aufruf der Funktion KDCADMI

Die UTM-Administrationsfunktionen, die an der Programmschnittstelle zur Administration zur Verfügung gestellt werden, werden über die Funktion KDCADMI aufgerufen. Beim Aufruf von KDCADMI können Sie Zeiger auf vier Datenbereiche an UTM übergeben. Das sind:

- der *Parameterbereich* (parameter_area)

  Im Parameterbereich geben Sie an, welche Operation UTM ausführen soll. Sie geben an, ob Sie sich über Objekte bzw. Betriebsparameter der Anwendung informieren, die Konfiguration um ein Objekt erweitern, Eigenschaften von Objekten modifizieren oder ein Objekt löschen wollen.
  Soll die angeforderte Operation für ein bestimmtes Objekt oder eine Gruppe von Objekten durchgeführt werden, dann müssen Sie im Parameterbereich den Objekttyp des Objektes angeben.
  Nach Durchführung bzw. Anstoßen der Operation hinterlegt UTM den Returncode, der über Erfolg oder Misserfolg des Aufrufs informiert, und die Länge der zurückgelieferten Daten im Parameterbereich.

- der *Identifikationsbereich* (identification_area)

  Den Identifikationsbereich benötigen Sie zur Angabe von Objektnamen, wenn z.B. Objekte aus der Konfiguration gelöscht, Objekteigenschaften geändert oder Objekteigenschaften ausgegeben werden sollen. In diesen Fällen übergeben Sie im Identifikationsbereich alle Daten, die UTM benötigt, um die zu administrierenden Objekte eindeutig zu identifizieren.

- der *Selektionsbereich* (selection_area)

  Im Selektionsbereich können Sie bei der Abfrage von Informationen (siehe Operationscode KC_GET_OBJECT) Selektionskriterien an UTM übergeben. UTM liefert dann nur Informationen zu den Objekten zurück, für die diese Selektionskriterien zutreffen. Beispiel: Informationen zu allen Benutzern, die zur Zeit bei der Anwendung angemeldet sind.

- der *Datenbereich* (data_area)

  Im Datenbereich können Sie einerseits Informationen an UTM übergeben, die UTM zur Ausführung der Operation benötigt, z.B. beim Eintragen neuer Objekte in die Konfiguration den Namen und die Eigenschaften des neuen Objektes.

  Andererseits liefert UTM im Datenbereich die angeforderten Informationen an das Programm zurück, z.B. bei der Ausgabe von Objekteigenschaften.

11.1.1 KDCADMI-Funktionsaufruf

Ein C-Programm, das KDCADMI-Aufrufe absetzt, muss immer eine #*include*-Anweisung auf die Include-Datei bzw. das Include-Element *kcadminc.h* enthalten. In *kcadminc.h* ist die Funktion KDCADMI folgendermaßen deklariert:

```
void KDCADMI(struct kc_adm_parameter *  ,  /*    parameter_area       */
             void                   *  ,  /*    identification_area  */
             void                   *  ,  /*    selection_area       */
             void                   * );  /*    data_area            */
```

Die Funktion KDCADMI rufen Sie wie folgt auf:

```
KDCADMI(&parameter_area,
        &identification_area,
        &selection_area,
        &data_area );
```

Dabei ist:

&parameter_area

Adresse des Parameterbereichs *parameter_area*.

&identification_area

Adresse des Identifikationsbereichs *identification_area*.

&selection_area

Adresse des Selektionsbereichs *selection_area*.

&data_area

Adresse des Datenbereichs *data_area*.

Wird einer der vier Bereiche bei einem Aufruf nicht benötigt, dann muss als Bereichsadresse der Nullpointer übergeben werden.

11.1.2 Beschreibung der zu versorgenden Datenbereiche

Im Folgenden werden die Parameter- und Datenbereiche, die Sie bei einem KDCADMI-Aufruf an UTM übergeben können, allgemein beschrieben.

Genauere Angaben darüber, wie Identifikations-, Selektions-, Datenbereich und die Felder des Parameterbereichs bei den einzelnen Operationen zu belegen sind, finden Sie im Abschnitt „Operationscodes von KDCADMI".

In der folgenden Beschreibung bedeutet:

--> Es handelt sich um ein Eingabefeld. In diesem Feld übergeben Sie Information an UTM.

<-- Es handelt sich um ein Ausgabefeld. In diesem Feld liefert UTM Information an das Administrationsprogramm zurück.

Parameterbereich: parameter_area

Im Parameterbereich geben Sie an, welche Operation UTM durchführen soll. Dazu stehen die Felder *opcode*, *subopcode1* und *subopcode2* zur Verfügung. Im Feld *obj_type* legen Sie den Objekttyp des zu administrierenden Objektes fest.

Nach der Bearbeitung hinterlegt UTM im Parameterbereich den Returncode und die Länge der zurückgelieferten Daten. Am Returncode können Sie ablesen, ob der Aufruf erfolgreich war oder nicht.

Der Parameterbereich ist wie folgt durch die Struktur *kc_adm_parameter* festgelegt:

struct kc_adm_parameter

```
int version;
KC_ADM_RETCODE retcode;
int version_data;
KC_ADM_OPCODE opcode;
KC_ADM_SUBOPCODE subopcode1;
KC_ADM_SUBOPCODE subopcode2;
KC_ADM_TYPE obj_type;
int obj_number;
int number_ret;
int id_lth;
int select_lth;
int data_lth;
int data_lth_ret;
```

Eingabefelder der Struktur *kc_adm_parameter* (im Folgenden gekennzeichnet mit -->), die nicht benötigt werden, müssen Sie grundsätzlich mit „binär null“ versorgen. Die Felder *version*, *version_data* und *opcode* müssen bei jedem Aufruf von KDCADMI versorgt werden.

Die Felder der Datenstruktur haben folgende Bedeutung:

-->version

bezeichnet die Version der Programmschnittstelle, die von dem Anwenderprogramm verwendet wird.

Die Version der Programmschnittstelle ist ein Kennzeichen für die Ausprägung der Programmschnittstelle und das Layout der bei dem Aufruf übergebenen Parameterbereiche.

Die Version der Programmschnittstelle müssen Sie bei jedem Aufruf von KDCADMI explizit angeben. Bisher ist als Version nur KC_ADMI_VERSION_1 definiert.

Wird die Ausprägung der Programmschnittstelle in einer Folgeversion geändert, dann wird die Version der Programmschnittstelle erhöht. Sind die Erweiterungen kompatibel und möchten Sie in der neuen openUTM-Version weiter die bisherige Programmschnittstelle nutzen, dann brauchen Sie Ihre Administrationsprogramme nicht anzupassen und können die Version der Schnittstelle weiterhin mit KC_ADMI_VERSION_1 versorgen. Möchten Sie, dass das Administrationsprogramm die neue Programmschnittstelle nutzt, müssen Sie Ihre Programme umstellen und in *version* die Version der Programmschnittstelle der aktuellen openUTM-Version angeben.

Die Schnittstelle wird jeweils über mehrere openUTM-Versionen sourcekompatibel gehalten.

<--retcode

Im Feld *retcode* gibt UTM den Returncode des Aufrufs zurück.

Es gibt allgemeine und Funktions-spezifische Returncodes.

Die allgemeinen Returncodes können bei allen Aufrufen auftreten. Sie sind im Abschnitt "Returncodes" beschrieben.

Die Funktions-spezifischen Returncodes können nur bei bestimmten Aufrufen der Programmschnittstelle auftreten und sind deshalb bei der Beschreibung des entsprechenden Aufrufs aufgelistet.

Ist der Parameterbereich nicht in seiner gesamten Länge zugreifbar, dann wird im KB-Rückgabebereich des Vorgangs, in dem der KDCADMI-Aufruf bearbeitet wird, der KDCS-Returncode KCRCCC mit '70Z', der Returncode KCRCDC mit 'A100' belegt. Der Vorgang wird mit PEND ER abgebrochen.

Beim Aufruf müssen Sie das Feld *retcode* mit der Konstanten KC_RC_NIL belegen.

-->version_data

Version der verwendeten Datenstrukturen.

Die Version der Datenstrukturen ist ein Kennzeichen für das Layout der verwendeten Datenstrukturen. *version_data* müssen Sie bei jedem Aufruf von KDCADMI explizit angeben. Für *version_data* sollte in openUTM V7.0 die Konstante KC_VERSION_DATA_11 verwendet werden.

KC_VERSION_DATA (ohne Suffix) bezeichnet immer jeweils die aktuelle Version der Datenstrukturen. Programme, die von der Source-Kompatibilität der Schnittstelle profitieren möchten, sollten die Konstante KC_VERSION_DATA nicht verwenden, sondern bei *version_data* immer die Versionskonstante KC_VERSION_DATA_xx von der Schnittstellenversion angeben, für die das Programm geschrieben wurde.

KC_VERSION_DATA_11 ist die für openUTM V7.0 gültige Version, KC_VERSION_DATA_10 bezeichnet z.B. die für openUTM V6.5 gültige Version.

Wird das Layout der Datenstrukturen objektkompatibel geändert, dann wird KC_VERSION_DATA nicht erhöht und die Teilprogramme sind in der neuen openUTM-Version ablauffähig.

Wird das Layout der Datenstrukturen bei einer openUTM-Version inkompatibel geändert, z.B. die Datenstrukturen um neue Felder erweitert und dadurch vergrößert, dann wird die Version der Datenstrukturen erhöht. Die Konstanten KC_VERSION_DATA bzw. KC_VERSION_DATA_11 werden in derselben Include-Datei wie die Datenstrukturen definiert. Da die Schnittstelle sourcekompatibel ist, müssen Teilprogramme in diesem Fall nur neu übersetzt werden.

-->opcode, subopcode1, subopcode2

In diesen Feldern geben Sie an, welche Aktion UTM ausführen soll. Das Feld *opcode* müssen Sie bei jedem Aufruf von KDCADMI versorgen. Es gibt an, welche Operation ausgeführt werden soll. In den Feldern *subopcode1* und *subopcode2* können Sie abhängig von *opcode* die Aktion näher spezifizieren, die ausgeführt werden soll.

Was Sie in *opcode* angeben müssen, damit eine bestimmte Operation ausgeführt wird, ist in der folgenden Tabelle zusammengefasst. Die mit (*) gekennzeichneten Operationscodes sind so genannte Standard-Operationen, die im Abschnitt „Objekt- und Parametertypen für Standard-Operationen" näher erläutert werden.

Funktion	**Wert von *opcode***
Das gesamte Anwendungsprogramm austauschen. *BS2000-Systeme:* Anwendungsteile im Common Memory Pool austauschen, die für den Austausch vorgemerkt sind. *Unix-, Linux- und Windows-Systeme:* In *subopcode1* geben Sie an, ob die nächsthöhere Version, die nächstniedrigere Version oder die aktuelle Version des Anwendungsprogramms geladen werden soll.	KC_CHANGE_APPLICATION
UTM-Dump erzeugen.	KC_CREATE_DUMP
Neue Objekte in die Konfiguration aufnehmen.	KC_CREATE_OBJECT (*)
KDCDEF-Steueranweisungen online erzeugen (inverser KDCDEF).	KC_CREATE_STATEMENTS
Objekt löschen, d.h. aus der Konfiguration herausnehmen	KC_DELETE_OBJECT (*)

Funktion	**Wert von *opcode***
RSA-Schlüsselpaar für die Verschlüsselung der Kommunikation mit Clients erzeugen, aktivieren, löschen oder auslesen.	KC_ENCRYPT
Informationen über Objekte und Anwendungsparameter abfragen. Über *subopcode1* und *subopcode2* steuern Sie Art und Umfang der Informationen.	KC_GET_OBJECT (*)
Nur bei UTM-Cluster-Anwendungen: Für alle oder einen einzelnen Benutzer, die/der noch an einer ausgefallenen Knoten-Anwendung als angemeldet vermerkt sind/ist oder die/der einen an die ausgefallene Knoten-Anwendungen gebundenen Vorgang haben/hat, ein erneutes Anmelden ermöglichen. Sperre der Cluster-User-Datei nach nicht ordnungsgemäß beendetem KDCDEF-Lauf wieder aufheben.	KC_LOCK_MGMT
Objekteigenschaften oder Anwendungsparameter ändern.	KC_MODIFY_OBJECT (*)
Nur bei UTM-Cluster-Anwendungen: TACs oder TAC-Queues und offene Asynchron-Vorgänge aus einer beendeten in eine laufende Knoten-Anwendung importieren.	KC_ONLINE_IMPORT
Transaktion zurücksetzen, die sich im Zustand PTC befindet.	KC_PTC_TA
Nur auf BS2000-Systemen: Nachricht an ein Dialog-Terminal oder an alle mit der Anwendung verbundenen Dialog-Terminals senden.	KC_SEND_MESSAGE
Anwendungslauf beenden. In *subopcode1* und *subopcode2* geben Sie an, wie die Anwendung beendet werden soll (Abbruch, Normales Ende). Für UTM-Cluster-Anwendungen geben Sie an, ob eine einzelne Knoten-Anwendung oder die komplette UTM-Cluster-Anwendung beendet werden soll.	KC_SHUTDOWN
Verbindungen zu Druckern aufbauen, für die Nachrichten vorliegen.	KC_SPOOLOUT
System-Protokolldatei SYSLOG administrieren. In *subopcode1* geben Sie an, welche Aktion durchgeführt werden soll.	KC_SYSLOG
IP-Adresse eines einzelnen oder aller Kommunikationspartner aktualisieren. Auf BS2000-Systemen müssen die Kommunikationspartner mit T-PROT=SOCKET generiert sein.	KC_UPDATE_IPADDR
Benutzer-Protokolldatei(en) auf die nächste Dateigeneration umschalten	KC_USLOG

Vom angegebenen *opcode* ist abhängig, welche Angaben Sie in den anderen Feldern des Parameterbereichs und im Identifikationsbereich, Selektionsbereich und Datenbereich machen müssen bzw. dürfen. Im Abschnitt „Operationscodes von KDCADMI" ist für jeden Operationscode (Wert von *opcode*) beschrieben, welche Operationen durchgeführt werden können und welche Angabe Sie dafür in

den Datenbereichen machen müssen, die Sie an UTM übergeben. Die Beschreibung erfolgt in alphabetischer Reihenfolge der Operationscodes.

-->obj_type

Im Feld *obj_type* ist der Objekttyp des Objektes anzugeben, das administriert werden soll, oder der Typ der Anwendungsparameter, die abgefragt oder verändert werden sollen.

Welchen Objekttyp bzw. Parametertyp Sie angeben dürfen, ist abhängig von der gewünschten Operation, also von den Angaben in den Feldern *opcode*, *subopcode1* und *subopcode2*.

Objekt- und Parametertypen für Standard-Operationen

Die folgenden beiden Tabellen enthalten die Objekt- und Parametertypen, die in UTM für die Standard-Operationen unterstützt werden. Standard-Operationen sind:

- Anzeigen
- Erzeugen
- Modifizieren
- Löschen

In der Spalte „opcode“ der Tabelle finden Sie die Operationscodes, bei denen der jeweilige Objekt-/Parametertyp angegeben werden kann. Folgende Abkürzungen werden verwendet:

- CRE für KC_CREATE_OBJECT (Erzeugen)
- DEL für KC_DELETE_OBJECT (Löschen)
- GET für KC_GET_OBJECT (Anzeigen)
- MOD für KC_MODIFY_OBJECT (Modifizieren)

Objekttypen

Objekttyp	**Wert von *obj_type***	**opcode**
Abstrakte Syntax für die Kommunikation über OSI TP	KC_ABSTRACT_SYNTAX	GET
OSI TP-Zugriffspunkte der lokalen Anwendung	KC_ACCESS_POINT	GET
Application Context für die Kommunikation über OSI TP	KC_APPLICATION_CONTEXT	GET
Namen der lokalen Anwendung, die mit KDCDEF generiert wurden (BCAMAPPL-Anweisung oder in MAX APPLINAME)	KC_BCAMAPPL	GET
Nur auf BS2000-Systemen: Namen der Character Sets (CHAR-SET Anweisung)	KC_CHARACTER_SET	GET
Namen und Eigenschaften einer Knoten-Anwendung in einer UTM-Cluster-Anwendung	KC_CLUSTER_NODE	GET, MOD
Verbindungen für die verteilte Verarbeitung über LU6.1	KC_CON	GET, CRE, DEL
Datenbankanschluss	KC_DB_INFO	GET, MOD
Nur auf BS2000-Systemen: Edit-Optionen für die Bildschirmausgabe im Zeilenmodus	KC_EDIT	GET
Globale Sekundär-Speicherbereiche für KDCS-Teilprogramme zu Austausch von Daten zwischen Vorgängen (GSSB)	KC_GSSB	GET
Namen und Eigenschaften der HTTP-Deskriptoren (HTTP_DESCRIPTOR Anweisung)	KC_HTTP_DESCRIPTOR	GET
Keysets der Anwendung. Keysets legen die Zugriffsberechtigungen von Clients und Benutzern auf Services und LTERM-Partner fest	KC_KSET	GET, MOD, CRE, DEL
Lademodule einer UTM-Anwendung auf BS2000-Systemen, Shared Objects/DLLs einer UTM-Anwendung auf Unix-, Linux- und Windows-Systemen	KC_LOAD_MODULE	GET, MOD
LPAP-Partner für den Anschluss von Partner-Anwendungen bei der verteilten Verarbeitung über LU6.1	KC_LPAP	GET, MOD

Objekttyp	**Wert von *obj_type***	**opcode**
Sessions für die verteilte Verarbeitung über LU6.1	KC_LSES	GET, MOD, CRE, DEL
Lokale Transaktionscodes für Services, die Partner-Anwendungen bei der verteilten Verarbeitung über LU6.1 oder OSI TP zur Verfügung stellen	KC_LTAC	GET, MOD, CRE, DEL
LTERM-Partner für den Anschluss von Clients und Druckern	KC_LTERM	CRE, DEL, GET, MOD
Benutzereigene Meldungsmodule	KC_MESSAGE_MODULE	GET
Nur auf BS2000-Systemen: Multiplexanschlüsse [1]	KC_MUX	GET, MOD
Associations zu Partner-Anwendungen bei der verteilten Verarbeitung über OSI TP	KC_OSI_ASSOCIATION	GET
Verbindungen für die verteilte Verarbeitung über OSI TP	KC_OSI_CON	GET, MOD
OSI-LPAP-Partner für den Anschluss von Partner-Anwendungen bei der verteilten Verarbeitung über OSI TP	KC_OSI_LPAP	GET, MOD
Transaktionen im Zustand PTC	KC_PTC	GET
Teilprogramme der UTM-Anwendung und VORGANG-Exits	KC_PROGRAM	CRE, DEL, GET
Clients und Drucker. Unter „Clients“ sind zusammengefasst: Terminals, UPIC-Clients, TS-Anwendungen	KC_PTERM	CRE, DEL, GET, MOD
Temporäre Queues	KC_QUEUE	GET
Belegung der UTM-Funktionstasten	KC_SFUNC	GET
Eigenschaften des Anmeldeverfahrens	KC_SIGNON	GET
IP-Subnetze	KC_SUBNET	GET
Transaktionscodes lokaler Services und TAC-Queues	KC_TAC	CRE, DEL, GET, MOD
TAC-Klassen der Anwendung	KC_TACCLASS	GET, MOD
LTERM-Pools der Anwendung	KC_TPOOL	GET, MOD

Objekttyp	**Wert von *obj_type***	**opcode**
Transfer- Syntax für die Kommunikation über OSI TP	KC_TRANSFER_SYNTAX	GET
Benutzerkennungen der Anwendung einschließlich ihrer Queues	KC_USER	CRE, DEL, GET, MOD
Benutzerkennungen der Anwendung einschließlich ihrer Queues (optimierter Zugriff für UTM-Cluster-Anwendungen)	KC_USER_FIX, KC_USER_DYN1, KC_USER_DYN2	GET

[1] Der Objekttyp kann zwar auf allen Systemen angegeben werden, die Verwendung ist jedoch nur auf BS2000-Systemen sinnvoll.

Parametertypen

Parametertyp	**Wert von *obj_type***	**opcode**
Aktuelle Statistikwerte über die Auslastung einer UTM-Cluster-Anwendung	KC_CLUSTER_CURR_PAR	GET, MOD
Eigenschaften einer UTM-Cluster-Anwendung (z.B. Name der Cluster-Filebase, Einstellungen für die Überwachung der Knoten-Anwendungen) sowie aktuelle Einstellungen (z.B. Anzahl der gestarteten Knoten-Anwendungen)	KC_CLUSTER_PAR	GET, MOD
Aktuelle Einstellung von Anwendungsparametern und Statistikwerte über die Auslastung der Anwendung	KC_CURR_PAR	GET, MOD
Parameter für die Diagnose und das UTM-Accounting	KC_DIAG_AND_ACCOUNT_PAR	GET, MOD
Daten zur dynamischen Konfiguration: Anzahl der existierenden und reservierten Objekte, d.h. die Gesamtzahl der Tabellenplätze in den einzelnen Objekttabellen, und die Anzahl der Objekte, die noch dynamisch konfiguriert werden können	KC_DYN_PAR	GET
Anwendungsname, KDCFILE-Name und Maximalwerte der Anwendung wie z.B. Größe des Cache, Größe und Anzahl der Speicherbereiche für KDCS-Teilprogramme und die maximal erlaubte Prozesszahl der Anwendung	KC_MAX_PAR	GET, MOD
Name, Typ und Format eines Benutzer-spezifischen Meldungsziels	KC_MSG_DEST_PAR	GET
Aktuelle Belegung des Pagepools	KC_PAGEPOOL	GET
Allgemeine Informationen über die generierten temporären Queues: Maximale Anzahl Queues, maximale Anzahl Nachrichten für eine Queue, Verhalten voller Queues.	KC_QUEUE_PAR	GET
Systemparameter: Art und Version des Betriebssystems, Name des Rechners und Basisdaten zur Anwendung (Anwendungsname, Anwendung mit oder ohne verteilte Verarbeitung usw.)	KC_SYSTEM_PAR	GET
Prozesszahlen der Anwendung: maximale und aktuelle Anzahl der Prozesse der Anwendung sowie der Prozesse, die für die Bearbeitung von Asynchron-Aufträgen und für die Bearbeitung von Teilprogrammläufen mit blockierenden Aufrufen zur Verfügung stehen	KC_TASKS_PAR	GET, MOD

Parametertyp	**Wert von *obj_type***	**opcode**
Timer der Anwendung	KC_TIMER_PAR	GET, MOD
Globale Werte für die verteilte Verarbeitung, mit Ausnahme der für die verteilte Verarbeitung definierten Timer	KC_UTMD_PAR	GET

Datenstrukturen für Objekt- und Parametertypen

Für jeden der zu den Standard-Operationen gehörenden Objekt- und Parametertypen steht in der Include-Datei *kcadminc.h* eine Datenstruktur zur Verfügung, in der Sie Objekteigenschaften bzw. Parameterwerte an UTM übergeben können oder von UTM zurück erhalten. Entsprechende Datenstrukturen gibt es auch für einige der Operationen, die nicht zu den Standard-Operationen gehören. Die Datenstrukturen sind im Abschnitt „Datenstrukturen zur Informationsübergabe" beschrieben. Die Namen der Datenstrukturen sind wie folgt aufgebaut:

Zum Objekt- bzw. Parametertyp „*TYP*“ gehört die Datenstruktur „*typ_*`str`“, also z.B. zu KC_USER gehört die Datenstruktur *kc_user_str*, zu KC_MAX_PAR *kc_max_par_str*.

Analoges gilt für die Nicht-Standard-Operationen. Z.B. gehört zum Opcode KC_APPLICATION_PAR die Datenstruktur *kc_application_par_str*.

-->obj_number

Anzahl der Objekte, für die die gewünschte Operation durchgeführt werden soll. In *obj_number* geben Sie z.B. an, über wieviele Objekte UTM bei einer Informationsabfrage (KC_GET_OBJECT) informieren soll.

<--number_ret

In *number_ret* liefert UTM die tatsächliche Anzahl der Objekte zurück, für die die Operation durchgeführt wurde.

-->id_lth

Im Feld *id_lth* müssen Sie die Länge des Identifikationsbereichs *identification_area* angeben, den Sie beim Aufruf übergeben.
Wird kein Identifikationsbereich übergeben, ist *id_lth*=0 anzugeben.

-->select_lth

Im Feld *select_lth* müssen Sie die Länge der Datenstruktur angeben, die Sie im Selektionsbereich *selection_area* an UTM übergeben.
Wird kein Selektionsbereich übergeben, ist *select_lth*=0 anzugeben.

-->data_lth

Im Feld *data_lth* müssen Sie die Länge des Datenbereichs *data_area* angeben, den Sie beim Aufruf übergeben bzw. in den UTM die Daten zurückliefern soll.
Werden keine Daten im Datenbereich übergeben, ist *data_lth*=0 anzugeben.

<--data_lth_ret

Im Feld *data_lth_ret* gibt UTM die tatsächliche Länge der im Datenbereich zurückgelieferten Daten an.

Identifikationsbereich: identification_area

Der Identifikationsbereich *identification_area* dient dazu, das Objekt zu identifizieren, das administriert werden soll. Alle Objekte sind innerhalb der Gruppe eines bestimmten Objekttyps durch ihren *Objektnamen* eindeutig identifiziert.

Zur Übergabe der Objektnamen legen Sie folgende Union über den Identifikationsbereich.

union kc_id_area

```
char kc_name2[2];
char kc_name4[4];
char kc_name8[8];
char kc_name32[32];
struct kc_triple_str triple;
struct kc_long_triple_str long_triple;
struct kc_ptc_id_str ptc_id;
```

Es ist abhängig vom Funktionsaufruf, ob ein Objekt im Identifikationsbereich eindeutig angegeben werden muss oder nicht.

Zur eindeutigen Identifikation müssen Sie die Objektnamen wie folgt angeben:

- bei den Objekttypen KC_CON und KC_PTERM müssen Sie im Unionelement *long_triple* vom Typ *kc_long_triple_str* als Objektname das Tripel *name, prozessorname, bcamappl-name* angeben. Dabei ist *name* der Name des Objekts (z.B. PTERM-Name), *prozessorname* der Name des Rechners, auf dem sich das Objekt befindet, und *bcamappl-name* der Name der lokalen Anwendung, über den die Verbindung zwischen Objekt und Anwendung aufgebaut wird.

  struct kc_long_triple_str

  ```
  char p_name[8];
  char pronam[64];
  char bcamappl[8];
  ```

- beim Objekttyp KC_MUX auf BS2000-Sytsemen müssen Sie im Unionelement *triple* vom Typ *kc_triple_str* als Objektname das Tripel *name, prozessorname, bcamappl-name* angeben. Dabei ist *name* der Name des Objekts, *prozessorname* der Name des Rechners, auf dem sich das Objekt befindet, und *bcamappl-name* der Name der lokalen

  Anwendung, über den die Verbindung zwischen Objekt und Anwendung aufgebaut wird. Dieses Tripel übergeben Sie in dem Unionelement *triple* vom Typ *kc_triple_str* an UTM.

  struct kc_triple_str

  ```
  char p_name[8];
  char pronam[8];
  char bcamappl[8];
  ```

- bei einem LTERM-Pool (Objekttyp KC_TPOOL) müssen Sie das LTERM-Präfix, aus dem die Namen der LTERM-Partner des LTERM-Pools erzeugt werden, als Objektname übergeben. Das LTERM-Präfix muss im Unionelement *kc_name8* an UTM übergeben werden.
- bei dem Objekttyp KC_TACCLASS müssen Sie im Unionelement *kc_name2* die Nummer der TAC-Klasse als Objektname übergeben, wenn der Funktionsaufruf für eine bestimmte TAC-Klasse gilt. Ansonsten geben Sie binär null an, d.h. der Aufruf gilt für alle TAC-Klassen.
- bei dem Objekttyp KC_DB_INFO müssen Sie im Unionelement *kc_name2* die Identifikation der Datenbank (*db_id*) als Objektname übergeben, wenn der Funktionsaufruf für eine bestimmte Datenbank gelten soll. *db_id* ist eine Ziffer und repräsentiert die Datenbanken in der Reihenfolge, wie sie im KDCDEF-Lauf generiert wurden.
- bei Lademodulen, Shared Objects, DLLs (Objekttyp KC_LOAD_MODULE) und Teilprogrammen (KC_PROGRAM) übergeben Sie den bei der Generierung festgelegten Namen im Unionelement *kc_name32.*
- bei dem Objekttyp KC_SFUNC (UTM-Funktionstasten) müssen Sie im Unionelement *kc_name4* die Kurzbezeichnung für die Funktionstaste als Objektname übergeben.
- bei der Funktion KC_PTC_TA (Rücksetzen einer Transaktion im Zustand PTC) müssen Sie das Unionelement *kc_ptc_id_str* mit den Werten aus der Struktur *ptc_ident* füllen. Den Inhalt von *ptc_ident* erhalten Sie durch einen vorherigen Aufruf KC_GET_OBJECT mit Objekttyp KC_PTC.

  Die Datenstruktur *kc_ptc_id_str* ist folgendermaßen definiert:

  struct kc_ptc_id_str

  ```
  char vg_indx[10];
  char vg_nr[10];
  char ta_nr_in_vg[5];
  ```

- bei den übrigen Objekttypen ist der bei der Generierung angegebene Name des Objekts im Unionfeld *kc_name8* zu übergeben, wenn der Funktionsaufruf für ein bestimmtes Objekt gilt. Ansonsten geben Sie binär null an, d.h. der Aufruf gilt für alle Objekte dieses Typs.

Wird der Identifikationsbereich bei einem Aufruf nicht unterstützt, dann müssen Sie als Bereichsadresse den Nullpointer übergeben. Im Parameterbereich müssen Sie dann *id_lth*=0 setzen.

Selektionsbereich: selection_area

Im Selektionsbereich kann bei der Abfrage von Informationen (Operationscode KC_GET_OBJECT) eine Datenstruktur mit Selektionskriterien an UTM übergeben werden. UTM liefert dann nur Namen und Eigenschaften der Objekte des angegebenen Objekttyps zurück, die den Selektionskriterien entsprechen.

Die Selektionskriterien müssen Sie in der Datenstruktur übergeben, die in *kcadminc.h* für den Objekttyp (*obj_type*) definiert ist. In der Datenstruktur müssen Sie die Strukturfelder, nach denen selektiert werden soll, mit den gesuchten Werten besetzen.

Beispiel

> Sie wollen Informationen zu Benutzerkennungen abfragen, über die zur Zeit ein Benutzer oder Client angemeldet ist. Dazu legen Sie die Datenstruktur *kc_user_str* über den Selektionsbereich und geben im Feld *connect_mode* den Wert "Y" an.

Werden mehrere Selektionskriterien gleichzeitig angegeben, werden nur die Objekte geliefert, die alle Selektionskriterien erfüllen. Die restlichen Strukturfelder sind mit binär null zu versorgen. Nach welchen Kriterien selektiert werden kann, ist bei der Beschreibung von KC_GET_OBJECT im Abschnitt "KC_GET_OBJECT - Informationen abfragen" angegeben.

Wollen Sie Selektionskriterien übergeben, dann müssen Sie beim KDCADMI-Aufruf die Adresse des Selektionsbereichs übergeben und im Feld *select_lth* des Parameterbereichs die Länge der Datenstruktur angeben, die Sie im Selektionsbereich übergeben.

Wird der Selektionsbereich bei einem Aufruf nicht verwendet, müssen Sie als Bereichsadresse *&selection_area* den Nullpointer übergeben. Im Parameterbereich müssen Sie dann *select_lth*=0 setzen.

Datenbereich: data_area

Der Datenbereich wird zur Übergabe von Objekteigenschaften, Parameterwerten und Informationen an bzw. von UTM benutzt. Die Struktur der Daten ist abhängig vom Operationscode und vom Typ des zu administrierenden Objektes.

Werden beim KDCADMI-Aufruf im Datenbereich Daten an UTM übergeben, dann müssen Sie beim KDCADMI-Aufruf die Bereichsadresse des Datenbereichs übergeben und im Feld *data_lth* des Parameterbereichs die Länge der Datenstruktur angeben, die Sie im Datenbereich übergeben.

Werden Informationen angefordert, die UTM im Datenbereich zurückliefert, dann müssen Sie beim KDCADMI-Aufruf die Bereichsadresse des zur Verfügung gestellten Datenbereichs übergeben und im Feld *data_lth* des Parameterbereichs seine Länge angeben.

Wird der Datenbereich bei einem Aufruf nicht benötigt, dann müssen Sie den Nullpointer als Bereichsadresse übergeben. Im Parameterbereich müssen Sie dann *data_lth*=0 setzen.

Der Datenbereich darf maximal 16 MB groß sein.

11.1.3 Returncodes

Der Returncode von KDCADMI besteht aus einem Maincode und einem Subcode. Der Maincode gibt an, ob die gewünschte Funktion durchgeführt bzw. die Durchführung angestoßen wurde (Returncode KC_MC_OK), oder nicht durchgeführt werden konnte (Returncode ungleich KC_MC_OK). Der Subcode gibt, wenn er ungleich KC_SC_NO_INFO ist, weitere Auskunft zu dem angegebenen Maincode.

Der Returncode wird in der folgenden Datenstruktur zurückgeliefert:

```
typedef struct
{  KC_MAINCODE     mc;
   KC_SUBCODE      sc;
} KC_ADM_RETCODE;
```

UTM liefert den Returncode in *retcode* des Parameterbereichs zurück. Ist der Parameterbereich nicht in seiner gesamten Länge zugreifbar oder nicht auf Wortgrenze ausgerichtet, dann setzt UTM im Rückgabebereich des Kommunikationsbereichs (KB) den Returncode KCRCCC = 70Z und den Returncode KCRCDC = A100. Der Vorgang wird mit PEND ER abgebrochen.

Sowohl die Maincodes als auch die Subcodes sind in der Include-Datei als Aufzählungstyp (*enum*) definiert. KDCADMI liefert also eine Zahlenkonstante zurück.

Um bei Fehlern die Diagnose zu erleichtern, können Sie sich die Main- und Subcodes als Strings (z.B. "KC_MC_OK") auflisten lassen. Dazu müssen Sie in Ihrem Programm den symbolischen Namen KC_ADM_GEN_STRING mit einer #define - Anweisung definieren, bevor Sie *kcadminc.h* inkludieren:

```
#define KC_ADM_GEN_STRING
#include kcadminc.h
```

Allgemeine Returncodes (Operationscode-unabhängig)

In der folgenden Tabelle finden Sie die Returncodes, die bei jeder Operation mit KDCADMI (bei jedem Operationscode) auftreten können. Andere Returncodes treten nur im Zusammenhang mit bestimmten Operationscodes auf. Diese Returncodes sind in der folgenden Beschreibung der einzelnen Operationscodes aufgelistet.

Main code = KC_MC_OK Die Funktion wurde ausgeführt bzw. die Durchführung der Funktion veranlasst. **Subcode:**
KC_SC_NO_INFO

Maincode = KC_MC_VERS_DATA_NOT_SUPPORTED Im Feld *version_data* des Parameterbereichs wurde eine Version der Datenstrukturen angegeben, die UTM nicht unterstützt. **Subcode:**
KC_SC_NO_INFO

Maincode = KC_MC_VERSION_NOT_SUPPORTED

Im Feld *version* des Parameterbereichs wurde eine Version der Programmschnittstelle angegeben, die UTM nicht unterstützt.

Subcode:

KC_SC_NO_INFO

Maincode = KC_MC_AREA_INVALID

Einer der beim KDCADMI-Aufruf übergebenen Datenbereiche ist nicht in der benötigten Länge zugreifbar, z. B. weil die Bereichsadresse ungültig ist oder der Bereich nicht in der erforderlichen Länge allokiert ist.

Subcodes:

KC_SC_ID_AREA

Der Identifikationsbereich ist nicht in der benötigten Länge zugreifbar.

KC_SC_SEL_AREA

Der Selektionsbereich ist nicht in der benötigten Länge zugreifbar.

KC_SC_DATA_AREA

Der Datenbereich ist nicht in der benötigten Länge zugreifbar oder
die Adresse des Parameterbereichs liegt innerhalb des Datenbereichs.

Maincode = KC_MC_NO_ADM_TAC

Der Transaktionscode, der den Administrationsaufruf abgesetzt hat, besitzt nicht die Berechtigung, die für die angeforderte Operation nötig ist (Administrationsberechtigung bzw. ADM-READ-Berechtigung).

Subcode:

KC_SC_NO_INFO

Maincode = KC_MC_PAR_INVALID

Im Parameterbereich wurde ein ungültiger Wert angegeben oder ein Feld wurde nicht gesetzt.

Subcodes:

KC_SC_RETCODE

Das Feld *retcode* des Parameterbereichs ist nicht mit KC_RC_NIL belegt.

KC_SC_OPCODE

Der in *opcode* des Parameterbereichs angegebene Operationscode ist ungültig.

KC_SC_SUBOPCODE1

Die in *subopcode1* des Parameterbereichs angegebene Operationsmodifikation ist ungültig.

KC_SC_SUBOPCODE2

Die in *subopcode2* des Parameterbereichs angegebene Operationsmodifikation ist ungültig.

KC_SC_TYPE

Der in *obj_type* des Parameterbereichs angegebene Objekttyp ist ungültig.

KC_SC_NUMBER

Die in *obj_number* des Parameterbereichs angegebene Anzahl von Objekten ist ungültig.

KC_SC_ID_LTH

Die in *id_lth* des Parameterbereichs angegebene Länge ist ungültig.

Mögliche Gründe:

- *id_lth* ist ungleich der Länge des Namensfelds für den Objekttyp.
- *id_lth* > 0, obwohl kein Identifikationsbereich übergeben werden darf.

KC_SC_SELECT_LTH

Die in *select_lth* des Parameterbereichs angegebene Länge ist ungültig.

Mögliche Gründe:

- *select_lth* ist ungleich der Länge der Datenstruktur des Objekttyps
- *select_lth* > 0, obwohl keine Selektion erlaubt ist

KC_SC_DATA_LTH

Die in *data_lth* des Parameterbereichs angegebene Länge ist ungültig.

Mögliche Gründe:

- *data_lth* ist ungleich der Länge der Datenstruktur des Objekttyps bzw. bei KC_GET_OBJECT kleiner als *obj_number* * Länge der Datenstruktur des Objekttyps.
- *data_lth* > 0, obwohl kein Datenbereich übergeben wird.
- *data_lth* > 16 MB

KC_SC_NUMBER_RET

number_ret des Parameterbereichs wurde nicht binär null gesetzt.

KC_SC_DATA_LTH_RET

data_lth_ret des Parameterbereichs ist nicht binär null.

Maincode = KC_MC_FUNCT_NOT_SUPPORTED Die angeforderte Operation wird in dem Betriebssystem, in dem die Anwendung abläuft, bzw. in der Version des Betriebssystems nicht unterstützt. Diesen Returncode liefert UTM z.B. zurück, wenn in einer UTM-Anwendung auf Unix-, Linux- oder Windows-Systemen eine Operation durchgeführt werden soll, die nur für UTM-Anwendungen auf BS2000-Systemen definiert ist. **Subcode:**
KC_SC_NO_INFO

11.1.4 Versorgung der Datenstrukturfelder bei der Datenübergabe

Die Felder der Datenstrukturen, die im Identifikationsbereich, Selektionsbereich und Datenbereich zur Datenübergabe zwischen UTM und den Administrationsprogrammen verwendet werden, bestehen nur aus dem Datentyp „char“. Die eckigen Klammern nach den Feldnamen beinhalten die Länge des Feldes. Fehlt die eckige Klammer, dann ist das Feld ein Byte lang.

Bei der Übergabe von Daten zwischen einem Administrationsprogramm und UTM ist Folgendes zu beachten:

- Namen und Schlüsselwörter müssen linksbündig abgelegt und rechts mit Leerzeichen aufgefüllt werden. Die Übergabe der Daten an UTM muss, außer bei Objektnamen, in Großbuchstaben erfolgen.

  Objektnamen können auch in Kleinbuchstaben angegeben werden. Es erfolgt jedoch keine Umsetzung in Großbuchstaben. Beim Eintragen neuer Objekte mit KC_CREATE_OBJECT sind die Angaben im Abschnitt „Format und Eindeutigkeit der Objektnamen" zu beachten.

  Beispiel: Das Feld *ptype* (*kc_pterm_str*) ist 8 Byte lang. *ptype* =APPLI wird wie folgt abgelegt: APPLI*bbb* (*b* =Leerzeichen).
- Numerische Daten werden von UTM in den Feldern rechtsbündig abgelegt und mit führenden Leerzeichen zurückgeliefert. Bei der Übergabe numerischer Daten vom Administrationsprogramm an UTM werden links- und rechtsbündige Angaben akzeptiert. Rechtsbündige Angaben werden mit führenden Leerzeichen oder Nullen akzeptiert. Linksbündige Angaben können mit dem Null-Byte (\0) abgeschlossen (sofern die Größe des Feldes ausreicht) oder mit Leerzeichen aufgefüllt werden.

  Beispiel: Das Feld *conn_users* (*kc_max_par_str*) ist 10 Byte lang. *conn_users* =155 kann z.B. wie folgt übergeben werden (*b*=Leerzeichen):
  *'bbbbbbb*155' oder '0000000155' oder '155\0' oder '155*bbbbbbb'*
- Felder der Datenstrukturen, in denen kein Wert übergeben wird, sind mit binär null zu versorgen.

11.2 Operationscodes von KDCADMI

In diesem Abschnitt finden Sie eine Übersicht der Parameter, die Sie in Abhängigkeit von der durchzuführenden Operation an UTM übergeben müssen. Die Beschreibung ist in alphabetisch aufsteigender Reihenfolge nach den Operationscodes gegliedert, die Sie im Feld *opcode* des Parameterbereichs übergeben.

Format der Beschreibung

Die Beschreibung zu einem Operationscode besteht aus vier Teilen:

1. Der erste Teil umfasst eine allgemeine Beschreibung der ausführbaren Aktionen, eine Auflistung der Voraussetzungen, die erfüllt sein müssen, damit UTM die gewünschte Aktion ausführen kann, und Hinweise auf Besonderheiten, die bei der Durchführung der Aktion zu beachten sind.

   Bei Änderungen der Konfiguration und der Eigenschaften werden Aussagen zur Wirkungsdauer der ausgeführten Änderungen gemacht sowie darüber, ob die Wirkung für UTM-Cluster-Anwendungen global oder lokal ist.

   Ist die beschriebene Administrationsfunktion oder ein Teil der Funktion auch von einem Administrationskommando (Transaktionscode von KDCADM) ausführbar, dann wird mit folgendem Symbol auf das Kommando verwiesen:

2. Eine Tabelle, in der kurz dargestellt wird, welche Bereiche (Parameter-, Identifikations-, Selektions- bzw. Datenbereich) bei den einzelnen Aktionen zu versorgen sind und welche Angaben in diesen Bereichen gemacht werden müssen.
3. Eine grafische Darstellung des Aufrufs mit allen notwendigen und möglichen Angaben und den Rückgaben von UTM. In den Grafiken sind die Felder grau gerastert, die Sie vor dem Aufruf der Funktion versorgen müssen. Alle Felder des Parameterbereichs, die hier nicht aufgeführt sind, sind vor dem KDCADMI-Aufruf mit binär null zu versorgen.

   In den Tabellen bedeutet „—“, dass in diesem Bereich keine Daten an UTM übergeben werden müssen.
4. Erläuterungen zur Grafik, d.h. zu den erforderlichen Angaben und den Rückgaben von UTM.

11.2.1 KC_CHANGE_APPLICATION - Anwendungsprogramm austauschen

Mit KC_CHANGE_APPLICATION können Sie während des Anwendungslaufs den Austausch des gesamten Anwendungsprogramms einleiten. Damit können ohne Beendigung der Anwendung Teilprogramme ersetzt und neue Teilprogramme zum Anwendungsprogramm hinzugefügt werden. Zum Programmaustausch siehe auch openUTM-Handbuch „Einsatz von UTM-Anwendungen“.

Mit KC_CHANGE_APPLICATION können Sie folgende Aktionen durchführen:

- Eine UTM-Anwendung auf einem BS2000-System, die mit Lademodulen generiert ist, in allen Prozessen beenden und neu laden.

  Diese Funktion benötigen Sie, um Lademodule in einem Common Memory Pool auszutauschen. Beim Neuladen wird die jeweils aktuelle Version der Lademodule geladen, die zuvor mit einem KC_MODIFY_OBJECT-Aufruf für den Objekttyp KC_LOAD_MODULE vorgemerkt wurde.

  Das Beenden des Anwendungsprogramms in allen Prozessen mit anschließendem Neuladen bewirkt außerdem, dass alle Lademodule entladen werden, die mit Lademodus ONCALL generiert sind.

  Es sind nur *subopcode1*=KC_NEW und KC_SAME möglich. KC_SAME hat die gleiche Wirkung wie KC_NEW.
- Das gesamte Anwendungsprogramm einer UTM-Anwendung auf Unix-, Linux- und Windows-Systemen kann durch das Anwendungsprogramm der nächsthöheren Dateigeneration ersetzt werden (*subopcode1*=KC_NEW), die im Dateigenerationsverzeichnis PROG (im Basisverzeichnis *filebase* der Anwendung) steht.

  Sie können mit KC_CHANGE_APPLICATION den Programmaustausch auch rückgängig machen, d.h. wieder auf das zuvor geladene Anwendungsprogramm zurückschalten (*subopcode1*=KC_OLD), oder das Anwendungsprogramm neu laden (*subopcode1*=KC_SAME), ohne auf eine andere Dateigeneration umzuschalten.

Folgende Voraussetzungen müssen erfüllt sein:

- Ist eine UTM-Anwendung auf einem BS2000-System mit Lademodulen generiert, müssen Sie die Teile der Anwendung, die in einem Common Memory Pool liegen und ausgetauscht werden sollen, zuvor mit KC_MODIFY_OBJECT-Aufrufen Objekttyp KC_LOAD_MODULE (siehe "obj_type=KC_LOAD_MODULE") zum Austausch vormerken lassen.
- Für den Austausch eines UTM-Anwendungsprogramms auf Unix-, Linux- und Windows-Systemen sollten die verschiedenen Versionen des Anwendungsprogramms (auch das aktuell geladene) mit Hilfe des UTM-Tools KDCPROG im Dateigenerationsverzeichnis *filebase*/PROG bzw. *filebase*\PROG verwaltet werden. Das Dateigenerationsverzeichnis muss mit Hilfe von KDCPROG erstellt worden sein (KDCPROG CREATE). Falls das Dateigenerationsverzeichnis nicht existiert, lädt UTM das Anwendungsprogramm *filebase*/`utmwork` (auf Unix- und Linux-Systemen) bzw. *filebase*\`utmwork` (auf Windows-Systemen) neu.

Der Programmaustausch ist im openUTM-Handbuch „Einsatz von UTM-Anwendungen“ beschrieben.

Beim Austausch des Anwendungsprogramms ist Folgendes zu beachten:

- Im neuen Anwendungsprogramm hinzugefügte Teilprogramme müssen bei der KDCDEF-Generierung definiert oder durch die Administration dynamisch konfiguriert worden sein.
- Im neuen Anwendungsprogramm sollten keine Teilprogramme fehlen, die vorher vorhanden waren. Aufträge, die für einen Transaktionscode angenommen wurden, für den nach einem Programmaustausch kein Teilprogramm mehr da ist, werden von UTM bei der Ausführung abnormal beendet (PEND ER).

Ablauf / Wirkungsdauer / Transaktionssicherung / Cluster

Durch den Aufruf wird der Programmaustausch angestoßen, d.h. ein Auftrag zum Programmaustausch erzeugt. Bei Rückkehr in das Teilprogramm ist der Austausch noch nicht durchgeführt. Der Programmaustausch unterliegt nicht der Transaktionssicherung. Er kann nicht durch einen in derselben Transaktion folgenden RSET-Aufruf zurückgesetzt werden.

Der Austausch des Anwendungsprogramms wird für jeden Prozess der Anwendung einzeln durchgeführt, indem das laufende Anwendungsprogramm für diesen Prozess beendet und das neue Anwendungsprogramm geladen wird. Zu einem Zeitpunkt wird das Anwendungsprogramm nur in einem Prozess ausgetauscht, um zu vermeiden, dass der laufende Betrieb durch den Programmaustausch erheblich gestört wird. In der Zeit, in der das Anwendungsprogramm in einem Prozess ausgetauscht wird, werden Aufträge von den anderen Prozessen bearbeitet. Diese Aufträge können dann auch Prozesse erhalten, in denen noch das alte Anwendungsprogramm abläuft. Dadurch werden in der Austauschphase Aufträge entweder noch vom alten oder schon vom neuen Anwendungsprogramm bearbeitet.

In UTM-Cluster-Anwendungen gilt (Unix-, Linux- und Windows-Systeme):
Der Aufruf wirkt Cluster-global, d.h. ein Anwendungsaustausch wird in jeder laufenden Knoten-Anwendung angestoßen.

Nach der Bearbeitung des Auftrags informiert UTM Sie mit einer Meldung über Erfolg bzw. Misserfolg des Programmaustausches. UTM erzeugt die Meldung K074, wenn der Programmaustausch erfolgreich durchgeführt werden konnte. Konnte UTM den Programmaustausch nicht durchführen, wird die Meldung K075 erzeugt. Treten Fehler auf, dann wird zusätzlich zu K074 bzw. K075 die Meldung K078 ausgegeben, die als Insert die Fehlerursache enthält.

KDCAPPL ("KDCAPPL - Eigenschaften und Grenzwerte für den Betrieb ändern"), Operand PROG

Versorgung der zu übergebenden Bereiche

Funktion des Aufrufs	***Angabe im***			
	Parameterbereich[1]	***Identifikations-bereich***	***Selektions-bereich***	***Datenbereich***
In UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen mit Shared Objects/DLLs: Aktuelles Anwendungsprogramm gegen das Anwendungsprogramm der nächsthöheren Version austauschen.	*subopcode1: KC_NEW*	——	——	—— *(Beim Aufruf muss der Zeiger auf einen Datenbereich für die Rückgaben von UTM übergeben werden.)*
In UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen mit Shared Objects/DLLs: Programmaustausch rückgängig machen, d.h. das aktuell geladene Anwendungsprogramm gegen das Anwendungsprogramm der nächstniedrigeren Version austauschen.	*subopcode1: KC_OLD*	——	——	
In UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen mit Shared Objects/DLLs: Anwendungsprogramm aus derselben Dateigeneration neu laden.	*subopcode1: KC_SAME*	——	——	
In UTM-Anwendungen auf BS2000-Systemen mit Lademodulen: *Das Anwendungsprogramm in allen Prozessen beenden und neu starten, um Anwendungsteile im Common Memory Pool auszutauschen. Statische Anwendungsteile lassen sich damit auch austauschen, wenn man die Anwendung zuvor neu bindet.*	*subopcode1: KC_NEW / KC_SAME*	——	——	——

[1] *In allen Fällen muss im Parameterbereich der Operationscode KC_CHANGE_APPLICATION angegeben werden.*

Versorgung der Parameter	
Parameterbereich	
Feldname	*Inhalt*
version	*KC_ADMI_VERSION_1*
retcode	*KC_RC_NIL*
version_data	*KC_VERSION_DATA_11*
opcode	*KC_CHANGE_APPLICATION*
subopcode1	*KC_NEW / KC_SAME / KC_OLD (Unix-, Linux- und Windows-Systeme)*
id_lth	*0*
select_lth	*0*
data_lth	*Länge des Datenbereichs / 0*
Identifikationsbereich	
—	
Selektionsbereich	
—	
Datenbereich	
—	

KDCADMI-Aufruf
KDCADMI (&parameter_area, NULL, NULL, &data_area)

Rückgaben von UTM	
Parameterbereich	
Feldname	*Inhalt*
retcode	*Returncodes*
data_lth_ret	*tatsächliche Länge der Daten im Datenbereich*
Datenbereich	
Datenstruktur kc_change_application_str / —	

subopcode1

Mit subopcode1 legen Sie fest, welche Art des Programmaustauschs durchgeführt werden soll. Folgende Angaben sind möglich:

KC_NEW *Beim Programmaustausch einer UTM-Anwendung auf Unix-, Linux- oder Windows-Systemen lädt UTM das Anwendungsprogramm aus der nächsthöheren Dateigeneration.*

Bei einer UTM-Anwendung auf einem BS2000-System, die mit Lademodulen generiert ist, beendet UTM das Anwendungsprogramm nacheinander in allen Prozessen und lädt es sofort wieder. Dabei wird die jeweils aktuelle Version der Lademodule geladen, d.h. die durch KC_MODIFY_OBJECT-Aufrufe vorgemerkten Lademodule im Common Memory Pool werden ausgetauscht.

Statische Anwendungsteile lassen sich damit auch austauschen, wenn man die Anwendung zuvor neu bindet.

KC_OLD *Beim Programmaustausch einer UTM-Anwendung auf Unix-, Linux- oder Windows-Systemen lädt UTM das Anwendungsprogramm aus der nächstniedrigeren Dateigeneration.*

Damit kann wieder das alte Anwendungsprogramm geladen werden, wenn nach Umschalten auf eine neue Dateigeneration Fehler im Anwendungsprogramm entdeckt werden.

KC_SAME *Auf Unix-, Linux- und Windows-Systemen lädt openUTM das Anwendungsprogramm aus der derselben Dateigeneration.*

Auf BS2000-Systemen wirkt KC_SAME wie KC_NEW.

data_lth

Im Feld data_lth geben Sie die Länge des Datenbereichs an, der für die Rückgaben von UTM zur Verfügung stehen soll.

Beim Programmaustausch einer UTM-Anwendung auf Unix-, Linux- oder Windows-Systemen müssen Sie data_lth >= sizeof (kc_change_application_str) setzen.

Beim Aufruf der Funktion müssen Sie den Zeiger auf den Datenbereich übergeben.

Beim Programmaustausch einer UTM-Anwendung auf einem BS2000-System, die mit Lademodulen generiert ist, können Sie data_lth=0 angeben. UTM liefert dann keine Daten zurück.

retcode

Im Feld retcode liefert UTM den Returncode des Aufrufs zurück. Neben den im Abschnitt „Returncodes" aufgelisteten Returncodes können beim Austausch des Anwendungsprogramms zusätzlich folgende Returncodes auftreten:

Maincode = KC_MC_REJECTED *Der Aufruf wurde von UTM abgewiesen.* ***Subcodes:***
KC_SC_NOT_CHANGEABLE *Die Anwendung wurde im Dialog gestartet. Ein Programmaustausch ist nicht möglich.*
KC_SC_FILE_ERROR (nur auf Unix-, Linux- und Windows-Systemen) *Beim Zugriff auf die Dateigeneration des zu ladenden Anwendungsprogramms ist ein Fehler aufgetreten. UTM hat die Meldung K043 mit dem DMS-Returncode erzeugt.*
KC_SC_NOT_GEN (nur auf BS2000-Systemen) *Die UTM-Anwendung ist ohne Lademodule generiert.*
KC_SC_NO_GLOB_CHANG_POSSIBLE *Nur bei UTM-Cluster-Anwendungen:* *Keine globalen Administrationsänderungen möglich, da die Generierung der Knoten-Anwendungen zur Zeit nicht konsistent ist.*
KC_SC_JFCT_RT_CODE_NOT_OK *Nur bei UTM-Cluster-Anwendungen:* *UTM-interner Fehler.* *Bitte wenden Sie sich an die Systembetreuung.*

Maincode = KC_MC_REJECTED_CURR

Der Aufruf kann zur Zeit nicht bearbeitet werden.

Subcode:

KC_SC_CHANGE_RUNNING

Es läuft gerade ein Programmaustausch, d.h. ein zuvor gestarteter Programmaustausch ist noch nicht abgeschlossen.

KC_SC_INVDEF_RUNNING

Nur bei UTM-Cluster-Anwendungen:
Es läuft gerade ein inverser KDCDEF, d.h. der Auftrag kann z.Zt. nicht bearbeitet werden.

Maincode = KC_MC_RECBUF_FULL

Nur bei UTM-Cluster-Anwendungen.

Subcode:

KC_SC_NO_INFO

Der Puffer mit Wiederanlauf-Information ist voll. (Siehe openUTM-Handbuch „Anwendungen generieren“, KDCDEF-Steueranweisung MAX, Parameter RECBUF).

data_lth_ret

Im Feld data_lth_ret des Parameterbereichs liefert UTM die tatsächliche Länge der Daten im Datenbereich zurück.

Datenbereich

Beim Programmaustausch einer UTM-Anwendung auf Unix-, Linux- oder Windows-Systemen liefert UTM im Datenbereich die Datenstruktur kc_change_application_str zurück, sofern beim Aufruf von KDCADMI der Zeiger auf einen Datenbereich übergeben wurde.

struct kc_change_application_str
`char program_fgg_new[4];` `char program_fgg_old[4];`

program_fgg_new

In diesem Feld liefert UTM die Dateigenerationsnummer des Anwendungsprogrammes zurück, das durch den Programmaustausch geladen wird.

program_fgg_old

In diesem Feld liefert UTM die Dateigenerationsnummer des Anwendungsprogrammes zurück, das vor dem Programmaustausch geladen war.

11.2.2 KC_CREATE_DUMP - UTM-Dump erzeugen

Mit KC_CREATE_DUMP können Sie einen UTM-Dump für Diagnosezwecke erzeugen (mit REASON=DIAGDP), ohne dass der Anwendungslauf abgebrochen wird.

Der Dump wird von dem Prozess erzeugt, in dem der KDCADMI-Aufruf abgesetzt wurde.

Ablauf / Wirkungsdauer / Transaktionssicherung / Cluster

Der Aufruf unterliegt nicht der Transaktionssicherung. Er wirkt unmittelbar. Die Aktionen, die durch den Aufruf ausgelöst werden, sind bei der Rückkehr in das Teilprogramm bereits ausgeführt.

Für UTM-Cluster-Anwendungen gilt (Unix-, Linux- und Windows-Systeme):
Der Aufruf wirkt Knoten-lokal, d.h. das Erzeugen eines UTM-Dumps für Diagnosezwecke wird nur in dieser Knoten-Anwendung ausgeführt.

KDCDIAG ("KDCDIAG - Diagnosehilfen ein- und ausschalten") Operand DUMP

Versorgung der zu übergebenden Bereiche

Funktion des Aufrufs	Angabe im			
	Parameterbereich	Identifikationsbereich	Selektionsbereich	Datenbereich
UTM-Dump erzeugen	KC_CREATE_DUMP	——	——	——

Versorgung der Parameter	
Parameterbereich	
Feldname	Inhalt
version	KC_ADMI_VERSION_1
retcode	KC_RC_NIL
version_data	KC_VERSION_DATA_11
opcode	KC_CREATE_DUMP
id_lth	0
select_lth	0
data_lth	0
Identifikationsbereich	
—	
Selektionsbereich	
—	

Datenbereich
—

KDCADMI-Aufruf
KDCADMI (&parameter_area, NULL, NULL, NULL)

Rückgaben von UTM	
Parameterbereich	
Feldname	Inhalt
retcode	Returncodes

UTM liefert nur die im Abschnitt „Returncodes" angegebenen Returncodes zurück.

11.2.3 KC_CREATE_OBJECT - Objekte in die Konfiguration eintragen

Mit KC_CREATE_OBJECT können Sie folgende Objekte dynamisch in die Konfiguration der Anwendung eintragen:

- Transportverbindungen zu entfernten LU6.1-Anwendungen (KC_CON)
- Keysets (KC_KSET)
- LU6.1-Sessions (KC_LSES)
- Transaktionscodes, über die Service-Programme in Partner-Anwendungen gestartet werden (KC_LTAC)
- LTERM-Partner zum Anschluss von Clients und Druckern (KC_LTERM)
- Anwenderteilprogramme und VORGANG-Exits (KC_PROGRAM)
- Clients und Drucker (KC_PTERM)
- Transaktionscodes und TAC-Queues (KC_TAC)
- Benutzerkennungen einschließlich ihrer Queues (KC_USER)

> openUTM auf Windows-Systemen unterstützt keine Drucker.

Pro KC_CREATE_OBJECT-Aufruf kann genau ein Objekt erzeugt werden. Innerhalb eines Teilprogramms kann KC_CREATE_OBJECT jedoch mehrmals aufgerufen werden, d.h. es können mehrere Objekte gleichen oder verschiedenen Objekttyps erzeugt werden.

Detaillierte Angaben zum dynamischen Eintragen von Objekten in die Konfiguration finden Sie im Kapitel „Konfiguration dynamisch ändern".

> Wenn bei **UTM-Cluster-Anwendungen** (Unix-, Linux- und Windows-Systeme) ein dynamisch erzeugbares Objekt gelöscht werden soll, müssen Sie dies grundsätzlich per Administration löschen. Diese Objekte können nicht durch eine Neugenerierung allein gelöscht werden.

Voraussetzungen für das dynamische Eintragen eines Objektes:

- Bei der KDCDEF-Generierung der UTM-Anwendung wurden mit RESERVE Tabellenplätze für den Objekttyp des Objektes reserviert und einer dieser Tabellenplätze ist noch nicht belegt. Mit KC_GET_OBJECT und Parametertyp KC_DYN_PAR können Sie ermitteln, ob noch Tabellenplätze für den entsprechenden Objekttyp frei sind.
- Anwenderteilprogramme und VORGANG-Exits können Sie nur dynamisch eintragen, wenn die Anwendung mit Lademodulen (BS2000-Systeme) bzw. Shared Objects/DLLs (Unix-, Linux- und Windows-Systeme) generiert wurde. Das Teilprogramm bzw. der VORGANG-Exit muss mit einem Compiler erzeugt werden, für den bei der KDCDEF-Generierung bereits ein Teilprogramm statisch konfiguriert worden ist (PROGRAM-Anweisung).

  Nur auf BS2000-Systemen: Bei ILCS-fähigen Compilern genügt die statische Generierung eines Teilprogramms mit COMP=ILCS.
- Transaktionscodes für Teilprogramme, die eine X/Open-Programmschnittstelle nutzen, können nur dynamisch eingetragen werden, wenn bei der KDCDEF-Generierung mindestens ein Transaktionscode für ein X/Open Teilprogramm konfiguriert wurde.
-

Benutzerkennungen können nur dynamisch konfiguriert werden, wenn die Anwendung mit Benutzerkennungen generiert ist.

Hinweis für BS2000-Systeme:

- Benutzerkennungen mit Ausweiskarte können Sie nur dynamisch eintragen, wenn bei der KDCDEF-Generierung explizit Tabellenplätze für Benutzerkennungen mit Ausweiskarte reserviert worden sind und noch einer dieser Plätze frei ist.
- Benutzerkennungen mit Kerberos-Authentisierung können Sie nur dynamisch eintragen, wenn bei der KDCDEF-Generierung explizit Tabellenplätze für Benutzerkennungen mit Kerberos-Authentisierung reserviert worden sind und noch einer dieser Plätze frei ist.

Beim Eintragen neuer Objekte ist Folgendes zu beachten:

Beim Eintragen von Objekten, die miteinander in Beziehung stehen, müssen bestimmte Regeln eingehalten werden. Diese Regeln sind im Kapitel „Konfiguration dynamisch ändern" beschrieben. In Beziehung zueinander stehen beispielsweise:

- Transaktionscodes zu den ihnen zugeordneten Teilprogrammen und VORGANG-Exits,
- Clients/Drucker zu den zugehörigen LTERM-Partnern und den Verbindungs-Benutzerkennungen bzw. den Benutzerkennungen für das automatische KDCSIGN,
- Keysets, die von Benutzerkennungen, LTERM-Partnern und Transaktionscodes referenziert werden.

Ablauf / Wirkungsdauer / Transaktionssicherung / Cluster

Der Aufruf unterliegt der Transaktionssicherung. Auf ein dynamisch erzeugtes Objekt kann vor dem Transaktionsende nur innerhalb der eigenen Transaktion zugegriffen werden. Der Anwendungs-weite Zugriff ist erst nach Abschluss der Transaktion möglich. Insbesondere ist die Administration des Objektes erst nach Transaktionsende möglich (auch die Abfrage von Informationen). Innerhalb derselben Transaktion kann auf das Objekt nur beim Eintragen weiterer Objekte zugegriffen werden, die mit ihm in Beziehung stehen.

Der Aufruf wirkt über das Ende des aktuellen Anwendungslaufs hinaus. D.h. die dynamisch eingetragenen Objekte sind auch in folgenden Anwendungsläufen Bestandteil der Konfiguration (sofern sie nicht wieder gelöscht werden).

In UTM-Cluster-Anwendungen gilt (Unix-, Linux- und Windows-Systeme):
Der Aufruf wirkt Cluster-global, d.h. in allen Knoten-Anwendungen werden die Objekte dynamisch in die Konfiguration eingetragen.

Versorgung der zu übergebenden Bereiche

Funktion des Aufrufs	Angabe im			
	Parameterbereich[1]	**Identifikations-bereich**	**Selektions-bereich**	**Datenbereich**
Transportverbindungen zur entfernten LU6.1-Anwendung in die Konfiguration aufnehmen	*obj_type*: KC_CON	——	——	Datenstruktur *kc_con_str* mit Namen und Eigenschaften des Partners und der Verbindung
Keyset in die Konfiguration aufnehmen	*obj_type*: KC_KSET	——	——	Datenstruktur *kc_kset_str* mit Namen und Eigenschaften des Keysets
LU6.1-Session in die Konfiguration aufnehmen	*obj_type*: KC_LSES	——	——	Datenstruktur *kc_lses_str* mit Namen und Eigenschaften der beteiligten Partner
Transaktionscode, über den Service-Programme in Partner-Anwendungen gestartet werden, in die Konfiguration aufnehmen	*obj_type*: KC_LTAC	——	——	Datenstruktur *kc_ltac_str* mit Namen und Eigenschaften des LTACs und des Partners
LTERM-Partner in die Konfiguration aufnehmen	*obj_type*: KC_LTERM	——	——	Datenstruktur *kc_lterm_str* mit Namen und Eigenschaften des LTERM-Partners

Funktion des Aufrufs	Angabe im			
	Parameterbereich[1]	**Identifikations-bereich**	**Selektions-bereich**	**Datenbereich**
Teilprogramm oder VORGANG-Exit in die Konfiguration aufnehmen	*obj_type*: KC_PROGRAM	——	——	Datenstruktur *kc_program_str* mit Namen und Eigenschaften des Teilprogramms bzw. VORGANG-Exits
Client/Drucker (PTERM) in die Konfiguration aufnehmen	*obj_type*: KC_PTERM	——	——	Datenstruktur *kc_pterm_str* mit Namen und Eigenschaften des Client/Druckers
Transaktionscode oder TAC-Queue in die Konfiguration aufnehmen	*obj_type*: KC_TAC	——	——	Datenstruktur *kc_tac_str* mit Namen und Eigenschaften des Transaktionscodes bzw. der TAC-Queue
Benutzerkennung (einschließlich Queue) in die Konfiguration aufnehmen	*obj_type*: KC_USER	——	——	Datenstruktur *kc_user_str* mit Namen und Eigenschaften der Benutzerkennung und Queue

[1] In allen Fällen muss im Parameterbereich der Operationscode KC_CREATE_OBJECT angegeben werden.

Versorgung der Parameter	
Parameterbereich	
Feldname	Inhalt
version	KC_ADMI_VERSION_1
retcode	KC_RC_NIL
version_data	KC_VERSION_DATA_11
opcode	KC_CREATE_OBJECT
obj_type	Objekttyp
obj_number	1
id_lth	0
select_lth	0
data_lth	Länge der Daten im Datenbereich
Identifikationsbereich	
—	
Selektionsbereich	
—	
Datenbereich	
Datenstruktur des Objekttyps	

KDCADMI-Aufruf
KDCADMI (&parameter_area, NULL, NULL, &data_area)

Rückgaben von UTM	
Parameterbereich	
Feldname	Inhalt
retcode	Returncodes

obj_type
: Im Feld *obj_type* müssen Sie den Typ des Objektes angeben, das erzeugt werden soll. Folgende Objekttypen können Sie angeben:
KC_CON, KC_KSET, KC_LSES, KC_LTAC, KC_LTERM, KC_PROGRAM, KC_PTERM, KC_TAC, KC_USER.

obj_number
: Pro Aufruf kann nur ein Objekt erzeugt werden, deshalb ist *obj_number*=1 zu setzen.

data_lth
: Im Feld *data_lth* geben Sie die Länge der Datenstruktur an, die Sie im Datenbereich an UTM übergeben.

Datenbereich
: Im Datenbereich müssen Sie eine Datenstruktur mit dem Namen des neuen Objektes und den Eigenschaften übergeben, die diesem Objekt zugeordnet werden sollen. Für jeden einzelnen Objekttyp steht eine eigene Datenstruktur zur Verfügung, die Sie über den Datenbereich legen müssen.

  In den folgenden Tabellen ab Abschnitt "obj_type = KC_CON" sind die Datenstrukturen abhängig vom Objekttyp des zu erzeugenden Objektes beschrieben. Sie können den Tabellen entnehmen, welche Felder in der jeweiligen Datenstruktur zu versorgen sind. In den Tabellen bedeutet die Auszeichnung in der 1. Spalte Folgendes:

  - o Versorgung des Feldes ist optional
  - m die Versorgung des Feldes ist Pflicht (mandatory)
  - (m) abhängig von den Angaben, die Sie für andere Pflichtparameter gemacht haben, oder vom Betriebssystem, in dem die UTM-Anwendung abläuft, kann die Versorgung des Feldes Pflicht sein

  Felder der Datenstrukturen, die Sie nicht explizit versorgen, müssen mit binär null vorbesetzt werden. Für diese Felder werden von UTM die Standardwerte eingesetzt. Die Standardwerte finden Sie bei der Beschreibung der Datenstrukturen im Abschnitt „Datenstrukturen zur Informationsübergabe".

retcode
: Im Feld *retcode* liefert UTM den Returncode des Aufrufs zurück, siehe „Returncodes".

11.2.3.1 obj_type = KC_CON

Um eine neue LU6.1-Transportverbindung zu einer entfernten Anwendung zu erzeugen, müssen Sie die Datenstruktur *kc_con_str* über den Datenbereich legen.

Der folgenden Tabelle können Sie entnehmen, wie die Felder der Struktur zu versorgen sind.

	Feldname[1]	**Bedeutung**
m	co_name[8]	Name der Partner-Anwendung, mit der über die logische Verbindung kommuniziert werden soll. Zum Format des Namens siehe im Abschnitt „Format und Eindeutigkeit der Objektnamen".
		BS2000-Systeme: *co_name* kann entweder der BCAM-Name einer UTM-Partner-Anwendung (bei homogener Kopplung) oder der Name einer TRANSIT-Anwendung (bei heterogener Kopplung) sein.
		Unix-, Linux- und Windows-Systeme: Bei *co_name* müssen Sie den T-Selektor angeben, mit dem sich die Partner-Anwendung beim Transportsystem anmeldet. Das erste Zeichen muss ein Buchstabe sein.
m[2] o[3]	pronam_long[64]	Name des Partner-Rechners.
		Für *pronam_long* ist der Name des Prozessors anzugeben, auf dem die Partner-Anwendung *co_name* abläuft. Das ist der Name eines Unix-, Linux-, Windows- oder BS2000-Systems. Angegeben werden muss der vollständige Rechnername (FQDN), unter dem der Rechner im DNS bekannt ist. Der Name darf maximal 64 Zeichen lang sein. Anstelle eines bis zu 64 Zeichen langen FQDN-Namens kann weiterhin ein kurzer, maximal 8 Zeichen langer lokaler Name (im BS2000-System: BCAM-Name) des Partnerrechners angegeben werden. In diesem Fall muss der lokale Name unter Zuhilfenahme von externer Zusatzinformation (im BS2000-System: FQDN-Datei, im Unix-/Linux-/Windows-System: hosts-Datei) vom Transportsystem auf einen FQDN-Namen bzw. eine IP-Adresse abbildbar sein.
o	bcamappl[8]	bezeichnet einen Namen der lokalen Anwendung, wie er bei der Generierung in der Steueranweisung MAX oder BCAMAPPL festgelegt wurde. Es darf kein BCAMAPPL-Name angegeben werden, für den T-PROT=SOCKET generiert ist.
		Standard: Wird keine Angabe gemacht, so gilt der primäre Anwendungsname in MAX ...,APPLINAME=.

	Feldname[1]	Bedeutung	
m	lpap[8]	Name des LPAP-Partners der Partner-Anwendung, zu der die Verbindung aufgebaut werden soll. Der Name des LPAP-Partners, über den sich die Partneranwendung anschließt, muss bei der Generierung mit der Anweisung LPAP definiert worden sein.	
		Durch das Erzeugen mehrerer CON-Objekte mit gleichem LPAP- Namen werden parallele Verbindungen zur Partner-Anwendung konfiguriert. Dabei müssen Sie darauf achten, dass die parallelen Verbindungen zu derselben Partner-Anwendung (*co_name* und *pronam*) führen.	
o	termn[2]	Maximal 2 Zeichen langes Kennzeichen für die Art des Kommunikationspartners. *termn* wird nicht von UTM abgefragt, es wird vom Benutzer zur Auswertung gesetzt, um beispielsweise Terminaltypen abzufragen oder zu gruppieren etc. Das Kennzeichen *termn* wird im KB-Kopf für Auftragnehmer-Vorgänge eingetragen, d.h. für Vorgänge, die von einer Partner-Anwendung in der lokalen Anwendung gestartet wurden.	
o[3]	listener_port [5]	Portnummer der Partner-Anwendung.	
		BS2000-Systeme: Eine Portnummer ungleich 0 darf nur angegeben werden, wenn die im Parameter *bcamappl* angegebene lokale Anwendung nicht mit T-PROT=NEA generiert ist.	
o[3]	t_prot	*Nur auf Unix, Linux- und Windows-Systemen:* enthält das Adressformat, mit dem sich die Partner-Anwendung beim Transportsystem anmeldet. Das Adressformat wird wie folgt angegeben:	
		'R'	RFC1006, ISO-Transportprotokoll Klasse 0 über TCP/IP und Konvergenzprotokoll RFC1006.
o[3]	tsel_format	*Nur auf Unix, Linux- und Windows-Systemen:* enthält den Formatindikator des T-Selektors der Partneradresse:	
		'T'	TRANSDATA-Format
		'E'	EBCDIC-Zeichenformat
		'A'	ASCII-Zeichenformat
		Zur Bedeutung der Adressformate siehe „Dokumentation zu PCMX" (openUTM-Dokumentation) .	

[1] Alle nicht aufgeführten Felder der Datenstruktur *kc_con_str* und alle für das verwendete Betriebssystem nicht relevanten Felder sind mit binär null zu versorgen. Die Datenstruktur ist im Abschnitt "kc_con_str - LU6.1-Verbindungen" vollständig beschrieben.

[2] Pflicht auf BS2000-Systemen.

[3] Optional auf Unix-, Linux- und Windows-Systemen.

11.2.3.2 obj_type = KC_KSET

Um ein neues Keyset zu erzeugen, müssen Sie die Datenstruktur *kc_kset_str* über den Datenbereich legen. Der folgenden Tabelle können Sie entnehmen, wie die Felder der Struktur zu versorgen sind.

	Feldname[1]	**Bedeutung**	
m	ks_name[8]	Name des Keysets.	
o	master	gibt an, ob es sich um ein Master-Keyset handelt. Ein Master-Keyset enthält alle Key- oder Zugangscodes, die für den Zugriff auf die Objekte der Anwendung notwendig sind, d.h. alle Keycodes zwischen 1 und dem Maximalwert, der bei der KDCDEF-Generierung in MAX KEYVALUE festgelegt wurde.	
		'Y'	Das Keyset ist ein Master-Keyset.
		'N'	Das Keyset ist kein Master-Keyset.
o	keys[4000]	Mit diesem Feld werden die Key- oder Zugangscodes ausgewählt, die diesem Keyset zugeordnet werden sollen. Es dürfen nur Keys bis zum generierten Maximalwert (MAX KEYVALUE) ausgewählt werden. Für jeden Key, der in dem Keyset enthalten sein soll, muss das entsprechende Byte des Feldes auf 1 gesetzt werden; alle nicht ausgewählten *keys*-Felder müssen den Wert 0 enthalten. Soll z.B. der Key 10 erzeugt werden, dann muss keys[9] den Wert 1 enthalten (Beachte: Das Array beginnt mit Index 0). Bei 4000 Keys wird für den Recovery-Puffer eine Größe von mindestens 16500 Byte empfohlen (Generierungsanweisung MAX, Parameter RECBUF).	

[1] Alle nicht aufgeführten Felder der Datenstruktur *kc_kset_str* und alle für das verwendete Betriebssystem nicht relevanten Felder sind mit binär null zu versorgen. Die Datenstruktur ist im Abschnitt "kc_kset_str - Keysets der Anwendung" vollständig beschrieben.

11.2.3.3 obj_type = KC_LSES

Um eine neue LU6.1-Session zu erzeugen, müssen Sie die Datenstruktur *kc_lses_str* über den Datenbereich legen. Der folgenden Tabelle können Sie entnehmen, wie die Felder der Struktur zu versorgen sind.

	Feldname[1]	**Bedeutung**
m	ls_name[8]	ist der Name der Session innerhalb der lokalen Anwendung (lokaler Half-Session-Name).
		Der angegebene Name muss eindeutig sein und darf auch keinem weiteren Objekt der Namensklasse 2 zugeordnet sein. Siehe dazu auch Abschnitt „Format und Eindeutigkeit der Objektnamen" .
m	lpap[8]	Name des LPAP-Partners, welcher der Partner-Anwendung zugeordnet wird. *ls_name* wird für die Kommunikation mit der Partner-Anwendung benutzt, die in der lokalen Anwendung dem LPAP-Partner *lpap* zugeordnet ist.
o	rses[8]	ist der Name, der die Session in der entfernten Anwendung bezeichnet (remote Half-Session-Name). Der Name darf bis zu 8 Zeichen lang sein.

[1] Alle nicht aufgeführten Felder der Datenstruktur *kc_lses_str* und alle für das verwendete Betriebssystem nicht relevanten Felder sind mit binär null zu versorgen. Die Datenstruktur ist im Abschnitt "kc_lses_str - LU6.1-Sessions" vollständig beschrieben.

11.2.3.4 obj_type = KC_LTAC

Um einen neuen Transaktionscode zu erzeugen, über den Service-Programme in Partner-Anwendungen gestartet werden, müssen Sie die Datenstruktur *kc_ltac_str* über den Datenbereich legen. Der folgenden Tabelle können Sie entnehmen, wie die Felder der Struktur zu versorgen sind.

	Feldname[1]	**Bedeutung**	
m	lc_name[8]	Name eines lokalen Transaktionscodes für das ferne Service-Programm.	
o	lpap[8]	legt fest, zu welcher Partner-Anwendung das Service-Programm gehört. *lpap* enthält • den Namen des LPAP- oder OSI-LPAP-Partners, der der Partner-Anwendung zugeordnet ist, • oder den Namen eines Master-LPAP-Partners. Wird *lpap* nicht angegeben, so muss der Name der Partner-Anwendung im Funktionsaufruf APRO (im Feld KCPA) spezifiziert werden.	
o	rtac[64]	Der Name des zugehörigen Transaktionscodes in der fernen Anwendung (*recipient_TPSU_title*).	
o	rtac_lth[2]	gibt an, wie lang der Name *rtac* ist. Angegeben wird die Anzahl der relevanten Byte in *rtac*. Minimalwert: 1, Maximalwert: 64	
o	code_type	gibt an, welcher Codetyp von UTM intern für den *rtac*-Namen verwendet wird:	
		'I'	INTEGER Der TAC-Name in *rtac* ist eine ganze positive Zahl zwischen 0 und 67108863. *rtac*-Namen vom Codetyp INTEGER sind nur für Partner-Anwendungen zulässig, die keine UTM-Anwendungen sind und über das OSI TP-Protokoll kommunizieren.
		'P'	PRINTABLE-STRING Der TAC-Name in *rtac* ist als String angegeben. Er ist max. 64 Zeichen lang. Klein-/Großschreibung ist relevant. Ein TAC-Name mit Codetyp PRINTABLE-STRING kann folgende Zeichen enthalten: • A, B, C, . . . , Z • a, b, c, . . . , z • 0, 1, 2, . . . , 9 • die Sonderzeichen ' - : ? = , + . ( ) / (Leerzeichen)
		'T'	T61-STRING *rtac* enthält einen T61-String. Für den Codetyp T61-STRING unterstützt UTM alle Zeichen des Codetyps PRINTABLE-STRING und zusätzlich folgende Sonderzeichen: $ > < & @ # % ; * _

	Feldname[1]	**Bedeutung**	
o	state	legt fest, ob *lc_name* für das ferne Service-Programm nach Start der lokalen Anwendung gesperrt ist oder nicht.	
		'Y'	*c_name* ist nicht gesperrt. Aufträge für den zugehörigen fernen Service werden angenommen.
		'N'	*lc_name* ist gesperrt. Aufträge für den zugehörigen fernen Service werden nicht angenommen.
o	accesswait_sec[5]	Zeit in Sekunden, die nach dem Anfordern des fernen Services (Aufruf des LTACs) auf das Belegen einer Session (evtl. einschließlich Verbindungsaufbau) bzw. auf den Aufbau einer Association maximal gewartet wird.	
		Eine Wartezeit != 0 bedeutet bei Asynchron-Aufträgen (LTAC mit*ltac_type*='A'), dass der Auftrag immer in die lokale Message Queue für die Partner-Anwendung eingetragen wird. Dialog-Aufträge werden angenommen.	
		Eine Wartezeit *accesswait_sec*=0 bedeutet: Dialog-Aufträge werden abgewiesen, wenn zu dem Partner keine Session/Association generiert ist, für die die lokale Anwendung Contention Winner ist. Bei Asynchron-Aufträgen wird der FPUT-Aufruf mit einem Returncode abgewiesen, falls zur Partner-Anwendung keine logische Verbindung besteht. Besteht eine logische Verbindung zur Partner-Anwendung, dann wird die Nachricht in die lokale Message Queue eingetragen.	
		Dialog-Aufträge werden unabhängig vom Wert in *accesswait_sec* abgewiesen, wenn keine logische Verbindung zur Partner-Anwendung besteht. Gleichzeitig wird ein Verbindungsaufbau angestoßen.	
		Minimalwert: '0' (Aufträge werden abgewiesen) Maximalwert: '32767'	
o	replywait_sec[5]	Zeit in Sekunden, die UTM maximal auf die Antwort vom fernen Service wartet. Durch Begrenzung der Wartezeit kann gewährleistet werden, dass die Wartezeit für Clients bzw. Benutzer am Terminal nicht beliebig lang werden kann.	
		replywait_sec='0' bedeutet: Warten ohne Zeitbegrenzung.	
		Minimalwert: '0' Maximalwert: '32767'	
o	lock_code[4]	enthält den Lockcode, der dem fernen Service innerhalb der lokalen Anwendung zugeordnet ist (Zugriffsschutz). *lock_code* kann eine Zahl zwischen '0' und dem mit dem Operanden KEYVALUE der KDCDEF-Anweisung MAX definierten Maximalwert enthalten. '0' bedeutet, dass der LTAC nicht durch einen Lockcode geschützt ist.	
		lock_code schließt die Angabe von *access_list* aus.	

	Feldname[1]	Bedeutung	
o	ltac_type	gibt an, ob die lokale Anwendung mit dem fernen Service Aufträge im Dialog bearbeitet oder ob Asynchron-Aufträge an den Partner-Service übergeben werden.	
		'D'	Aufträge an den Partner-Service werden im Dialog bearbeitet.
		'A'	Der Partner-Service wird asynchron (über Message Queuing) gestartet.
o	ltacunit[4]	enthält die Anzahl der Verrechnungseinheiten, die in der Abrechnungsphase des UTM-Accounting für jeden Aufruf von *ltac* berechnet wird. Die Verrechnungseinheiten werden auf den Verrechnungseinheitenzähler der Benutzerkennung aufaddiert, die den *ltac* aufgerufen hat.	
		Minimalwert: '0', Maximalwert: '4095'	
o	access_list[8]	bezeichnet ein Keyset, das die Zugriffsrechte festlegt, die ein Benutzer der lokalen UTM-Anwendung haben muss, um einen Auftrag an das ferne Service-Programm senden zu dürfen. Ob der Auftrag in der fernen Anwendung ausgeführt wird, hängt auch von den dort definierten Zugriffsrechten ab. Das Keyset muss zuvor erzeugt oder bereits bei der Generierung definiert worden sein.	
		access_list schließt die Angabe von *lock_code* aus.	
		Ein Benutzer kann auf den LTAC nur dann zugreifen, wenn das Keyset des Benutzers, das Keyset des LTERM-Partners, über den der Benutzer angemeldet ist, und das angegebene Keyset mindestens einen gemeinsamen Keycode enthalten.	

[1] Alle nicht aufgeführten Felder der Datenstruktur *kc_ltac_str* und alle für das verwendete Betriebssystem nicht relevanten Felder sind mit binär null zu versorgen. Die Datenstruktur ist im Abschnitt "kc_ltac_str - Transaktionscodes ferner Services (LTAC)" vollständig beschrieben.

11.2.3.5 obj_type = KC_LTERM

Um einen neuen LTERM-Partner zu erzeugen, müssen Sie die Datenstruktur *kc_lterm_str* über den Datenbereich legen. Sie können keine LTERMs von Bündeln und Gruppen erzeugen.

Der folgenden Tabelle können Sie entnehmen, wie die Felder der Struktur zu versorgen sind.

	Feldname[1]	**Bedeutung**
m	lt_name[8]	Name des LTERM-Partners. Der Name darf bis zu 8 Zeichen lang sein. Er kann in Klein- oder Großbuchstaben angegeben werden und muss innerhalb der Namensklasse, zu der er gehört, eindeutig sein. Zum Format des Namens und zur Eindeutigkeit siehe Abschnitt „Format und Eindeutigkeit der Objektnamen". Namen von zuvor gelöschten LTERM-Partnern bzw. Transaktionscodes dürfen nicht verwendet werden.
o	kset[8]	Nur bei Dialog-Partnern (*usage_type*='D') relevant: Keyset der Anwendung, das dem LTERM-Partner zugeordnet werden soll. Das Keyset muss zuvor dynamisch erzeugt oder bereits bei der Generierung definiert worden sein. Ein Client kann einen mit Lockcode bzw. Access-List gesicherten Service nur starten, wenn sowohl im Keyset der Benutzerkennung, unter der sich der Client anmeldet, als auch im Keyset des zugehörigen LTERM-Partners der zum Lockcode bzw. der Access-List passende Key- bzw. Zugangscode enthalten ist.
		Hinweis: Wollen Sie in einer Anwendung, die mit Benutzerkennungen (USER) generiert ist, keinen Zugriffsschutz für LTERM-Partner definieren, dann ordnen Sie den LTERM-Partnern Keysets zu, die alle Keycodes der Anwendung enthalten (MASTER).
o	locale_lang_id[2] locale_terr_id[2] locale_ccsname[8]	*Nur auf BS2000-Systemen:* Legen die Sprachumgebung (Locale) des LTERM-Partners fest.
		In *locale_lang_id* geben Sie das Sprachkennzeichen der Landessprache an, in der Meldungen und Nachrichten an den LTERM-Partner übergeben werden sollen. Es ist maximal 2 Byte lang.
		In *locale_terr_id* geben Sie das Territorialkennzeichen an. Es kennzeichnet territoriale Besonderheiten in der Landessprache. Es ist maximal 2 Byte lang.
		In *locale_ccsname* geben Sie den CCS-Namen des erweiterten Zeichensatzes (**c**oded **c**haracter **s**et) an. Der CCS-Name ist bis zu 8 Bytes lang. Er muss zu einem auf dem BS2000-System definierten EBCDIC-Zeichensatz gehören, siehe XHCS-Benutzerhandbuch.
o	lock_code[4]	Nur bei Dialog-Partnern (*usage_type*='D') relevant: Lockcode, der dem LTERM-Partner zugeordnet werden soll (Zugangsschutz). Der Lockcode muss in dem Bereich liegen, der in der KDCDEF-Anweisung MAX im Operanden KEYVALUE definiert ist.

	Feldname[1]	Bedeutung	
o	state	Legt fest, ob der LTERM-Partner nach dem Erzeugen zunächst gesperrt sein soll oder nicht.	
		'Y'	Der LTERM-Partner soll nicht gesperrt werden. (ON)
		'N'	Der LTERM-Partner soll gesperrt werden. (OFF)
o	usage_type	Legt fest, ob der LTERM-Partner zum Anschluss von Dialog-Partnern oder zum Anschluss von Druckern konfiguriert wird:	
		'D'	LTERM-Partner zum Anschluss von Dialog-Partnern
		'O'	LTERM-Partner zum Anschluss eines Ausgabemediums, z.B. Drucker
o	user_gen[8]	Nur bei Dialog-Partnern (*usage_type*='D') relevant. Bei LTERM-Partnern von Terminals: Benutzerkennung, für die UTM beim Aufbau der logischen Verbindung ein automatisches KDCSIGN durchführen soll. Diese Benutzerkennung muss vor dem LTERM-Partner dynamisch oder statisch in die Konfiguration eingetragen worden sein. Bei LTERM-Partnern von UPIC-Clients und TS-Anwendungen: Die Verbindungs-Benutzerkennung muss in derselben Transaktion wie der LTERM-Partner erzeugt werden. Siehe dazu das Kapitel „Konfiguration dynamisch ändern".	
		Standard bei LTERM-Partnern von Terminals: Kein automatisches KDCSIGN Standard bei LTERM-Partnern von UPIC-Clients (*ptype*='UPIC-R') oder TS-Anwendungen (*ptype*='APPLI' oder 'SOCKET'): Verbindungs-Benutzerkennung mit dem Namen des LTERM-Partners.Wird diese Benutzerkennung nicht explizit in derselben Transaktion wie der LTERM-Partner erzeugt, dann erzeugt UTM diese Benutzerkennung implizit. Sie darf jedoch nicht vorher existieren.	
		Hinweis: Die Verwendung des automatischen KDCSIGN bei Terminals schränkt den Zugriffsschutz ein.	
o	cterm[8]	*Nur auf BS2000, Unix- und Linux-Systemen:* Nur für LTERM-Partner zum Anschluss von Druckern relevant (*usage_type*='O'). Name des Druckersteuer-LTERMs, über das der Drucker administriert werden soll. Das Druckersteuer-LTERM muss vor dem LTERM-Partner statisch oder dynamisch in die Konfiguration eingetragen worden sein (siehe im Kapitel „Konfiguration dynamisch ändern").	
		Jedem Drucker, der diesem LTERM-Partner zugeordnet wird (KC_PTERM), muss eine Drucker-Id (*cid*) zugeordnet werden, die im Bereich des Druckersteuer-LTERMs eindeutig ist.	

	Feldname[1]	**Bedeutung**
o	format_attr format_name[7]	*Nur auf BS2000-Systemen:* Nur relevant, wenn dem LTERM-Partner ein Terminal zugeordnet werden soll. Mit Hilfe dieser Felder können Sie dem LTERM-Partner ein LTERM-spezifisches Startformat zuweisen. Voraussetzung für das Zuweisen eines Startformats ist, dass ein Formatierungssystem generiert ist (KDCDEF-Anweisung FORMSYS). Ist das Startformat ein #Format, dann muss zusätzlich ein Anmelde-Vorgang generiert sein.
		Zur Definition eines Startformats müssen Sie immer *format_name* **und** *format_attr* angeben.
		In *format_attr* geben Sie das Formatkennzeichen des Startformats an:
		'A' für das Formatattribut ATTR (+Format). 'N' für das Formatattribut NOATTR (*Format). 'E' für das Formatattribut EXTEND (#Format).
		Bedeutung der Formatattribute siehe Abschnitt "kc_lterm_str - LTERM-Partner" ("format_attr, format_name (nur auf BS2000-Systemen)").
		In *format_name* geben Sie den Namen des Startformats an. Der Name kann bis zu 7 Zeichen lang sein und darf nur alphanumerische Zeichen enthalten.
o	plev[5]	*Nur auf BS2000, Unix- und Linux-Systemen:* Nur relevant bei LTERM-Partnern von Ausgabemedien (*usage_type* ='O'). In *plev* legen Sie den Schwellwert für die Message Queue des LTERM-Partners fest. Sobald in der Queue soviele Ausgabe-Aufträge stehen, wie in *plev* angegeben wurden, versucht UTM, die Verbindung zu dem Drucker automatisch aufzubauen. Ist dem LTERM-Partner ein Druckerbündel zugeordnet, dann baut UTM die Verbindung zu allen Druckern auf. UTM baut die Verbindung automatisch wieder ab, sobald die Message Queue leer ist.
		plev dürfen Sie nur zusammen mit qamsg='Y' angeben. *plev*='0' bedeutet, dass kein Schwellwert definiert wird.
		Minimalwert: '0' Maximalwert: '32767'

	Feldname[1]	Bedeutung	
o	qamsg	Legt fest, ob Asynchron-Aufträge (FPUT- und DPUT-Aufträge) an den Client/Drucker in der Message Queue des LTERM-Partners zwischengespeichert werden, auch wenn der Client /Drucker nicht mit der Anwendung verbunden ist.	
		'Y'	Ein Asynchron-Auftrag wird in die Message Queue eingetragen. *qamsg*='Y' ist nicht möglich bei *restart*='N'.
		'N'	Ein Asynchron-Auftrag wird abgewiesen, falls der zugehörige Client/Drucker nicht mit der Anwendung verbunden ist.
o	qlev[5]	Legt fest, wieviele Asynchron-Nachrichten maximal gleichzeitig in der Message Queue des LTERM-Partners zwischengespeichert werden. Wird der Schwellwert in *qlev* überschritten, dann weist UTM weitere Asynchron-Aufträge an diesen LTERM-Partner bzw. an den ihm zugeordneten Client/Drucker ab. Minimalwert:'0' Maximalwert:'32767'	
o	restart	Ist nur für Dialog-Partner relevant (LTERM-Partner mit *usage_type*='D') In *restart* legen Sie fest, wie UTM beim Verbindungsabbau mit Asynchron-Nachrichten verfährt, die zu diesem Zeitpunkt in der Message Queue des LTERM-Partners stehen.	
		'Y'	Asynchron-Nachrichten an den Client bleiben erhalten. In einer Anwendung ohne Benutzerkennungen führt UTM für diesen LTERM-Partner den automatischen Vorgangswiederanlauf durch. In einer UTM-Cluster-Anwendung ohne Benutzerkennungen ist 'Y' nur erlaubt, wenn sie mit CLUSTER USER-RESTART=YES generiert wurde.
		'N'	UTM löscht beim Verbindungsabbau alle Asynchron-Nachrichten aus der Queue. Handelt es sich dabei um Auftrags-Komplexe, so wird der negative Quittungs-Auftrag aktiviert. In einer Anwendung ohne Benutzerkennungen führt UTM für den LTERM-Partner keinen automatischen Vorgangswiederanlauf durch.
		Bei *qamsg*='Y' muss *restart*='Y' gesetzt sein.	

	Feldname[1]	Bedeutung	
o	annoamsg	*Nur auf BS2000-Systemen:* Ist nur für LTERM-Partner eines Terminals relevant. In *annoamsg* legen Sie fest, ob UTM eine Asynchron-Nachricht vor der Ausgabe am Terminal ankündigt:	
		'Y'	Asynchron-Nachrichten werden durch eine Meldung in der Systemzeile angekündigt.
		'N'	Asynchron-Nachrichten werden sofort (ohne Ankündigung) ausgegeben.
o	netprio	*Nur auf BS2000-Systemen:* Legt die Transportpriorität fest, die auf Transportverbindungen zwischen der Anwendung und dem Client/Drucker benutzt wird.	
		'M'	Transportpriorität „Medium“
		'L'	Transportpriorität „Low“
		Für native TCP/IP Verbindungen (*t_prot* = SOCKET) hat dieses Feld keine Bedeutung..	
o	kerberos_dialog	*Nur auf BS2000-Systemen:* Legt fest, ob beim Verbindungsaufbau für Clients, die Kerberos unterstützen und die über diesen LTERM-Partner direkt (nicht über OMNIS) mit der Anwendung verbunden sind, ein Kerberos-Dialog durchgeführt wird.	
		'Y'	Es wird ein Kerberos-Dialog durchgeführt.
		'N'	Es wird kein Kerberos-Dialog durchgeführt.

[1] Alle nicht aufgeführten Felder der Datenstruktur *kc_lterm_str* und alle für das verwendete Betriebssystem nicht relevanten Felder sind mit binär null zu versorgen. Die Datenstruktur ist im Abschnitt "kc_lterm_str - LTERM-Partner" vollständig beschrieben.

ℹ Die Zuordnung zwischen LTERM-Partner und Client/Drucker (LTERM zu PTERM) erfolgt beim Eintragen des Client/Drucker in die Konfiguration oder durch KC_MODIFY_OBJECT.

11.2.3.6 obj_type = KC_PROGRAM

Um ein neues Teilprogramm oder einen VORGANG-Exit in die Konfiguration aufzunehmen, müssen Sie die Datenstruktur *kc_program_str* über den Datenbereich legen.

Der folgenden Tabelle können Sie entnehmen, wie die Felder der Datenstruktur *kc_program_str* zu versorgen sind.

	Feldname[1]	**Bedeutung**
m	pr_name[32]	Name des Teilprogramms. Der Name darf bis zu 32 Byte lang sein. Bei der Namensangabe müssen Sie die Angaben im Abschnitt „Format und Eindeutigkeit der Objektnamen" beachten. Der Name eines zuvor aus der Konfiguration gelöschten Teilprogramms darf nicht verwendet werden. In UTM-Anwendungen auf BS2000-Systemen geben Sie den ENTRY- bzw. CSECT-Namen des Teilprogramms an.
(m)	compiler	Compiler bzw. ILCS-Fähigkeit des Compilers, mit dem das Teilprogramm übersetzt wurde.
		In UTM-Anwendungen auf BS2000-Systemen ist die Angabe von *compiler* Pflicht. Bei allen Teilprogrammen, die ILCS unterstützen, müssen Sie statt des Compilers 'I' für ILCS angeben.
		In UTM-Anwendungen auf BS2000-Systemen sind folgende Angaben möglich: 'I' für ILCS (Inter Language Communication Services) 'A' für den Assembler-Compiler ASSEMB 'C' für den C-Compiler (wird von UTM auf 'I' umgesetzt) '1' für den COBOL-Compiler (COB1) 'F' für den Fortran-Compiler (FOR1) 'X' für PASCAL-XT 'P' für PLI1 'S' für SPL4
		In einer UTM-Anwendung auf Unix-, Linux- und Windows-Systemen sind folgende Werte möglich: 'C' für den C-Compiler '+' für den C++-Compiler '2' für den COBOL-Compiler von Micro Focus '3' für den NetCOBOL-Compiler von Fujitsu

	Feldname[1]	Bedeutung
m	load_module[32]	Name des Lademoduls (BS2000-Systeme) bzw. des Shared Objects/derDLL (Unix-, Linux- und Windows-Systeme), in dem das Teilprogramm eingebunden ist. Der Name kann bis zu 32 Zeichen lang sein.
		BS2000-Systeme: Das Lademodul muss mit einer KDCDEF Steueranweisung LOAD-MODULE konfiguriert worden sein. Es darf nicht statisch zum Anwendungsprogramm gebunden sein.
		Unix-, Linux- und Windows-Systeme: Das Shared Object/DLL muss mit einer KDCDEF Steueranweisung SHARED-OBJECT statisch konfiguriert worden sein.

[1] Alle nicht aufgeführten Felder der Datenstruktur *kc_program_str* und alle für das verwendete Betriebssystem nicht relevanten Felder sind mit binär null zu versorgen. Die Datenstruktur ist im Abschnitt "kc_program_str - Teilprogramme und VORGANG-Exits" vollständig beschrieben.

11.2.3.7 obj_type = KC_PTERM

Um einen Drucker oder Client (d.h. ein Terminal, einen UPIC-Client oder eine TS-Anwendung) in die Konfiguration aufzunehmen, müssen Sie die Datenstruktur kc_pterm_str über den Datenbereich legen. In kc_pterm_str übergeben Sie Namen, Adresse und Eigenschaften des Client bzw. Druckers an UTM. Der folgenden Tabelle können Sie entnehmen, wie die Felder der Struktur zu versorgen sind.

> openUTM auf Windows-Systemen unterstützt keine Drucker.

	Feldname[1]	**Bedeutung**
m	pt_name[8]	Name des Client bzw. Druckers. Der Name darf bis zu 8 Zeichen lang sein. In *pt_name* ist der symbolische Name anzugeben, unter dem der Client/Drucker dem Transportsystem bekannt ist. Zum Format des Namens und zur Eindeutigkeit siehe im Abschnitt „Format und Eindeutigkeit der Objektnamen". Namen von zuvor gelöschten Objekten, die zu derselben Namensklasse gehören, dürfen nicht verwendet werden. Wenn Ihre Anwendung einen LTERM-Pool mit *connect_mode*='M' (Multi) enthält, dann darf das Tripel (*pt_name, pronam, bcamappl*) nicht mit einem Namenstripel des LTERM-Pools übereinstimmen (= Tripel aus dem Namen eines Pool-LTERM-Partners, dem Rechnernamen des Pool-Client und dem BCAMAPPL-Namen der Anwendung, über den die Verbindung vom Client aufgebaut wird). Es kann sich sonst kein Client mehr über diesen LTERM-Pool anschließen.
		Besonderheiten bei der Kommunikation über die Socket-Schnittstelle: Soll die Verbindung zwischen Kommunikationspartner und UTM-Anwendung über die Socket-Schnittstelle (SOCKET) erfolgen und soll der Partner beim Verbindungsaufbau eine bestimmte Portnummer verwenden, dann müssen Sie für *pt_name* den Wert PRT*nnnnn* angeben. Dabei ist *nnnnn* die Portnummer im fernen System, über die der Partner die Verbindung aufbaut. Ist der Partner eine UTM-Anwendung, dann ist die Angabe der Portnummer nicht möglich, da UTM die Portnummer nicht selbst setzt. Wird die Verbindung nicht von der Partner-Anwendung aufgebaut, sondern nur von der lokalen Anwendung aus, dann wird der Name nur intern gebraucht, z.B. für die Administration.

	Feldname[1]	**Bedeutung**
(m)	pronam_long[64]	Name des Rechners, auf/an dem sich der Client/Drucker befindet.
		Angegeben werden muss der vollständige Rechnername (FQDN), unter dem der Rechner im DNS bekannt ist. Der Name darf maximal 64 Zeichen lang sein. Anstelle eines bis zu 64 Zeichen langen FQDN-Namens kann weiterhin ein kurzer, maximal 8 Zeichen langer lokaler Name (im BS2000-System: BCAM-Name) des Partnerrechners angegeben werden. In diesem Fall muss der lokale Name unter Zuhilfenahme von externer Zusatzinformation (im BS2000-System: FQDN-Datei, im Unix-/Linux-/Windows-System: hosts-Datei) vom Transportsystem auf einen FQDN-Namen bzw. eine IP-Adresse abbildbar sein.
		Ist auf BS2000-Systemen *ptype*='*RSO', dann muss *pronam_long*='*RSO' angegeben werden.
		Wird die Verbindung zum Partner über die Socket-Schnittstelle aufgebaut, dann müssen Sie für *pronam_long* die symbolische Adresse bzw. den realen Hostnamen des Rechners angeben.
		Auf Unix-, Linux- und Windows-Systemen ist die Angabe von *pronam_long* nur für *ptype* ='UPIC-R', 'APPLI' oder 'SOCKET' erlaubt. Bei Terminals und Druckern setzt openUTM den Standardwert (Leerzeichen) ein.
o	bcamappl[8]	Name der UTM-Anwendung, über den die Verbindungen zwischen UTM-Anwendung und Client/Drucker aufgebaut werden sollen. Der Anwendungsname muss bei der KDCDEF-Generierung in MAX APPLINAME oder mit einer BCAMAPPL-Anweisung statisch generiert worden sein.
		Soll die Verbindung zum Kommunikationspartner über das Protokoll SOCKET aufgebaut werden, dann müssen Sie einen BCAMAPPL-Namen angeben, für den *t_prot* ='T' generiert ist.
		Nur auf BS2000-Systemen: Bei *ptype* ungleich 'APPLI', 'SOCKET' oder 'UPIC-R' darf für *bcamappl* nur der in MAX APPLINAME generierte Anwendungsname (Standardwert) angegeben werden.
(m)	ptype[8]	Typ des Client/Druckers. Eine Liste der möglichen Angaben finden Sie im Abschnitt "kc_pterm_str - Clients und Drucker"f. Bei *ptype*='APPLI', 'SOCKET' oder 'UPIC-R' muss *lterm* spezifiziert werden.
		Die Angabe von *ptype* ist in UTM-Anwendungen auf BS2000-Systemen Pflicht.
		Auf Windows-Systemen ist die Angabe von *ptype*='PRINTER' nicht erlaubt.
o	ptype_fotyp[8]	Ist nur bei Druckern auf Unix- und Linux-Systemen relevant (*ptype*='PRINTER'). In *ptype_fotyp* geben Sie den Druckertyp (*printertype*) an.

	Feldname[1]	**Bedeutung**	
o	ptype_class[40]	Ist nur bei Druckern auf Unix- und Linux-Systemen relevant (*ptype*='PRINTER'). In *ptype_class* geben Sie den Namen der Druckergruppe (printer class) an, zu der der Drucker gehört. Die Druckergruppe wird bei der Generierung auf Unix- und Linux-Systemen festgelegt.	
(m)	lterm[8]	Name des LTERM-Partners, der diesem Client/Drucker zugeordnet werden soll. Bei Terminals und Druckern ist die Angabe optional. Ihnen kann zu einem späteren Zeitpunkt durch die Administration ein LTERM-Partner zugeordnet werden. Wird in *lterm* der Name eines LTERM-Partners angegeben, dann muss er vor dem Terminal/Drucker statisch oder dynamisch in die Konfiguration eingetragen worden sein.	
		Bei UPIC-Clients und TS-Anwendungen (*ptype*= 'UPIC-L', 'UPIC-R', 'APPLI' oder 'SOCKET') ist *lterm* Pflichtparameter. Der angegebene LTERM-Partner muss in derselben Transaktion wie der Client erzeugt werden. Siehe dazu Kapitel „Konfiguration dynamisch ändern".	
o	auto_connect	Legt fest, ob die Verbindung zu dem Client/Drucker beim Start der Anwendung automatisch aufgebaut wird:	
		'Y'	Bei jedem Start der Anwendung soll UTM versuchen, die Verbindung zum Client /Drucker aufzubauen.
		'N'	Es soll kein automatischer Verbindungsaufbau durchgeführt werden.
		Für UPIC-Clients ist nur *auto_connect*='N' erlaubt.	
o	state	Legt fest, ob der Client/Drucker nach dem Eintragen zunächst gesperrt sein soll oder nicht.	
		'Y'	Der Client/Drucker soll nicht gesperrt werden. (ON)
		'N'	Der Client/Drucker soll gesperrt werden. (OFF)
o	cid[8]	Ist nur für Drucker auf BS2000-, Unix- und Linux-Systemen relevant. In *cid* geben Sie die Drucker-Id (CID) an. Die CID darf maximal 8 Zeichen lang sein. Die CID wird benötigt, wenn der Drucker über ein Druckersteuer-LTERM administriert werden soll. Das Druckersteuer-LTERM identifiziert den Drucker über die CID. Die CID muss im Bereich des Druckersteuer-LTERMs eindeutig sein.	

	Feldname[1]	Bedeutung
o	map	Nur relevant bei TS-Anwendungen (*ptype*='APPLI') oder SOCKET-USP-Anwendungen.
		In *map* legen Sie fest, ob UTM eine Code-Konvertierung (EBCDIC<->ASCII) für die Benutzer-Nachrichten, die zwischen den Kommunikationspartnern ausgetauscht werden, durchführen soll oder nicht. Benutzer-Nachrichten werden an der KDCS-Schnittstelle bei den Aufrufen zur Nachrichtenbehandlung (MPUT/FPUT/DPUT) im Nachrichtenbereich übergeben.
		'U' (USER) UTM konvertiert die Daten des Nachrichtenbereichs nicht, d.h. die Nachrichten werden unverändert zwischen den Kommunikationspartnern übertragen.
		'S' , '1' , '2', '3', '4' ist nur für folgende TS-Anwendungen erlaubt: • BS2000-Systeme: *ptype*='SOCKET' • Unix-, Linux- oder Windows-Systeme: *ptype*='APPLI' oder 'SOCKET' Geben Sie einen der obigen Werte an, dann konvertiert UTM die Benutzernachrichten gemäß den für die Code-Konvertierung bereitgestellten Konvertierungstabellen, siehe Abschnitt „Code-Konvertierung“ im openUTM-Handbuch „Anwendungen generieren“, d.h.: • Vor dem Senden wird auf Unix-, Linux- und Windows-Systemen von ASCII nach EBCDIC und auf BS2000-Systemen von EBCDIC nach ASCII konvertiert. • Nach dem Empfangen wird auf Unix-, Linux- und Windows-Systemen von EBCDIC nach ASCII und auf BS2000-Systemen von ASCII nach EBCDIC konvertiert. Die Angaben 'S' und '1 sind dabei synonym. Dabei geht UTM davon aus, dass die Nachrichten nur abdruckbare Zeichen enthalten. Weitere Informationen zur Code-Konvertierung finden Sie im openUTM-Handbuch „Anwendungen programmieren mit KDCS“, Abschnitt "Code-Konvertierung".
o	termn[2]	Kennzeichen für die Art des Client/Druckers (Terminalmnemonic). Das Kennzeichen ist maximal 2 Zeichen lang. Die Standardwerte für *termn* finden Sie in der Tabelle im Abschnitt "kc_pterm_str - Clients und Drucker" ("BS2000-Systeme" bzw. "Unix-, Linux- und Windows-Systeme").

	Feldname[1]	Bedeutung	
o	protocol	*Nur auf BS2000-Systemen:* Legt fest, ob auf Verbindungen zu dem Client/Drucker mit dem Benutzerdienstprotokoll NEABT gearbeitet werden soll.	
		'N' (NO): Es wird ohne NEABT gearbeitet.	
		'S' (STATION): Es wird mit NEABT gearbeitet.	
		Für Clients, die sich über einen Multiplexanschluss anschließen sollen, muss *protocol*='S' angegeben werden.	
		Für UPIC-Clients, RSO-Drucker und TS-Anwendungen, mit denen über die Socket-Schnittstelle kommuniziert werden soll, müssen Sie *protocol*='N' angeben. Die Angabe von *protocol*='S' wird in diesen Fällen ignoriert.	
o	usage_type	*Nur auf BS2000-Systemen:* Legt fest, ob ein Dialog-Partner oder ein Ausgabemedium konfiguriert wird. Folgendes können Sie angeben:	
		'D' für einen Dialog-Partner 'O' für ein Ausgabemedium (z.B. Drucker)	
o	listener_port[5]	Bei *listener_port* geben Sie an, an welcher Portnummer im fernen System die Partner-Anwendung auf Verbindungsaufbauwünsche von außen wartet. Es sind alle Portnummern erlaubt.	
		Auf BS2000-Systemen ist die Angabe von *listener_port* nur bei *ptype*='APPLI' oder 'SOCKET' erlaubt. Bei *ptype*='SOCKET' ist die Angabe Pflicht. Eine Portnummer ungleich 0 darf nur angegeben werden, wenn die bei *bcamappl* angegebene lokale Anwendung nicht mit T-PROT=NEA generiert ist.	
		Auf Unix-, Linux und Windows-Systemen ist *listener-port* nur für *t_prot*='T' und 'R' relevant.	
o	t_prot	nur relevant bei Clients vom Typ *ptype*='APPLI', 'SOCKET' oder 'UPIC-R' auf Unix-, Linux- und Windows-Systemen: In *t_prot* geben Sie das Adressformat der Transportadresse des Client an. Mögliche Werte sind:	
		'R'	RFC1006, ISO-Transportprotokoll Klasse 0 über TCP/IP und Konvergenzprotokoll RFC1006 (APPLI, UPIC-R)
		'T'	native TCP-IP-Transportprotokoll für die Kommunikation über die Socket-Schnittstelle (SOCKET)

	Feldname[1]	**Bedeutung**
o	tsel_format	nur relevant bei Clients vom Typ *ptype*='APPLI', 'SOCKET' oder 'UPIC-R' auf Unix-, Linux- und Windows-Systemen: In *tsel_format* geben Sie das Format des T-Selektors der Client-Adresse an. Mögliche Werte sind:
		'T' für das TRANSDATA-Format 'E' für das EBCDIC-Zeichenformat 'A' für das ASCII-Zeichenformat
o	idletime[5]	Darf nur für Dialog-Partner angegeben werden. In *idletime* legen Sie die Zeit in Sekunden fest, die UTM nach dem Ende einer Transaktion oder nach dem Anmelden (KDCSIGN) maximal auf eine Eingabe vom Client wartet. Bei Zeitüberschreitung wird die Verbindung zum Client abgebaut. Ist der Client ein Terminal, dann wird vor dem Verbindungsabbau die Meldung K021 ausgegeben. Der für *idletime* angegebene Wert darf nicht kleiner sein als die Timerangaben in *kc_timer_par_str.termwait_in_ta_sec* und *kc_timer_par_str.pgwttime_sec* (siehe im Abschnitt "kc_timer_par_str - Timer-Einstellungen" (pgwttime_sec)).
		Diese Funktion dient dazu den Datenschutz zu verbessern: Vergisst ein Benutzer bei einer Unterbrechung oder der Beendigung seiner Arbeit am Terminal, sich abzumelden, dann wird die Verbindung nach Ablauf der Wartezeit automatisch abgebaut. Damit wird die Wahrscheinlichkeit für einen unberechtigten Zugang verringert.
		Maximalwert: '32767' Minimalwert: '60' Der Wert 0 bedeutet Warten ohne Zeitbegrenzung. Bei Werten ungleich 0 und kleiner als 60 wird der Wert 60 verwendet.
		Bei einem ungültigen Wert setzt UTM *idletime* auf den kleinsten zulässigen Wert und liefert den Returncode KC_MC_OK mit Subcode KC_SC_INVALID_IDLETIME zurück.

	Feldname[1]	Bedeutung	
o	encryption_level	nur relevant für UPIC-Clients und auf BS2000-Systemen zusätzlich für einige Terminalemulationen.	
		In *encryption_level* legen Sie die minimale Verschlüsselungsebene für die Kommunikation mit dem Client fest, • ob die Verschlüsselung der Nachrichten standardmäßig angefordert wird oder nicht, • welche Verschlüsselungsebene dabei gefordert wird, • oder ob der Client ein „trusted“ Client ist. Folgende Angaben sind möglich:	
		'N'	(NONE) Die Verschlüsselung der Nachrichten wird von UTM **nicht** angefordert. Passworte werden verschlüsselt übertragen, sofern beide Partner Verschlüsselung unterstützen. Vorgänge, für deren Vorgangs-TACs Verschlüsselung generiert wurde (siehe *kc_tac_str.encryption_level* im Abschnitt "kc_tac_str - Transaktionscodes lokaler Services"), können von diesem Client nur gestartet werden, wenn der Client die Verschlüsselung aushandelt.
		'3'	(LEVEL 3) UTM fordert die Verschlüsselung der Nachrichten mit der Verschlüsselungsebene 3 an, d.h. die Nachrichten und Passworte werden mit dem AES-CBC Algorithmus verschlüsselt und zum Austausch des AES-Schlüssels wird ein RSA-Schlüssel mit einer Schlüssellänge von 1024 bit verwendet.
		'4'	(LEVEL 4) UTM fordert die Verschlüsselung der Nachrichten mit der Verschlüsselungsebene 4 an, d.h. die Nachrichten und Passworte werden mit dem AES-CBC Algorithmus verschlüsselt und zum Austausch des AES-Schlüssels wird ein RSA-Schlüssel mit einer Schlüssellänge von 2048 bit verwendet.
		'5'	(LEVEL 5) UTM fordert die Verschlüsselung der Nachrichten mit der Verschlüsselungsebene 5 an, d.h. die Nachrichten werden mit AES-GCM Algorithmus verschlüsselt. Die Vereinbarung des AES-Schlüssels erfolgt über das Ephemeral Elliptic Curve Diffie-Hellman Verfahren (ECDHE). Dabei wird zur Signatur des öffentlichen Server-Schlüssels ein RSA Schlüssel mit einer Schlüssellänge von 2048 Bits verwendet. Der Level 5 wird von openUTM nur für Unix-, Linux- und Windows-Systeme unterstützt.
			Der Verbindungsaufbau zu dem Client wird von UTM abgelehnt, wenn der Client nicht mindestens die angegebene Verschlüsselungsebene (3, 4 oder 5) unterstützt. *encryption_level*=3 ... 5 ist nur sinnvoll, wenn an Ihrem System die Verschlüsselungsfunktionen zur Verfügung stehen. Andernfalls kann sich der Client nicht anschließen.

		'T'	(TRUSTED) Der Client ist ein „trusted Client“. Nachrichten zwischen Client und Anwendung werden nicht verschlüsselt. Ein „trusted Client“ kann auch Vorgänge starten, deren Vorgangs-TACs Verschlüsselung erfordern (generiert mit *kc_tac_str.encryption_level*='2' oder '5'; siehe "kc_tac_str - Transaktionscodes lokaler Services"). Dieser Wert sollte nur gewählt werden, wenn der Client nicht für jedermann zugänglich ist und die Kommunikation über eine sichere Verbindung läuft.
		Für die einzelnen Client-Typen gilt bezüglich der Verschlüsselungsebene: • Für remote UPIC-Clients (PTYPE=UPIC-R) sind Verschlüsselungsebenen 3 bis 5 sinnvoll. • Bei lokalen UPIC-Clients (PTYPE=UPIC-L) einer Anwendung auf Unix-, Linux- und Windows-Systemen wird 3, 4 oder 5 von UTM durch TRUSTED ersetzt. • Für HTTP-Clients, die sich über einen Transportsystemendpunkt (BCAMAPPL) an die Anwendung anschließen, der mit T-PROT=(..., SECURE) generiert ist, wird von UTM als Encryption Level immer TRUSTED gesetzt. • Wird 3 ... 5 für einen Partner eines anderen Typs angegeben, dann wird der Wert von UTM durch in NONE ersetzt.	
		Voraussetzung für die Verschlüsselung der Daten auf Verbindungen zum Client ist, dass entsprechende RSA-Schlüssel verfügbar sind. Falls die Anwendung mit OPTION GEN-RSA-KEYS=NO generiert ist, dann erzeugt KDCDEF keine RSA-Schlüssel, d.h. es sind standardmäßig keine RSA-Schlüssel verfügbar. Es ist jedoch möglich, RSA-Schlüssel per KDCUPD zu übertragen oder mit KC_ENCRYPT zu erzeugen. Diese können dann von den neu erzeugten Objekten benutzt werden.	
o	usp_hdr	Dieser Parameter ist nur für PTERMs mit *ptype*='SOCKET' von Bedeutung. Er legt fest, für welche Ausgabe-Nachrichten UTM auf dieser Verbindung einen UTM-Socket-Protokoll-Header aufbaut. Mögliche Werte sind:	
		'A'	Bei allen Ausgabe-Nachrichten (Dialog, Asynchron, K-Meldungen) erzeugt UTM einen UTM-Socket-Protokoll-Header und stellt diesen der Nachricht voran.
		'M'	Nur bei Ausgabe von K-Meldungen erzeugt UTM einen UTM-Socket-Protokoll-Header und stellt diesen der Nachricht voran.
		'N'	UTM erzeugt für keine Ausgabe-Nachricht einen UTM-Socket-Protokoll-Header.

[1] Alle nicht aufgeführten Felder der Datenstruktur *kc_pterm_str* und alle für das verwendete Betriebssystem nicht relevanten Felder sind mit binär null zu versorgen. Die Datenstruktur ist im Abschnitt "kc_pterm_str - Clients und Drucker" vollständig beschrieben.

11.2.3.8 obj_type = KC_TAC

Um einen neuen Transaktionscode oder eine TAC-Queue zu erzeugen, müssen Sie die Datenstruktur kc_tac_str über den Datenbereich legen. Um eine TAC-Queue zu erzeugen, sind nur die folgenden Felder von Bedeutung: *tc_name, admin, qlev, q_mode, q_read_acl, q_write_acl, state* und *type*.

Alle anderen Felder werden für TAC-Queues nicht ausgewertet.

Der folgenden Tabelle können Sie entnehmen, wie die Felder der Datenstruktur *kc_tac_str* zu versorgen sind.

	Feldname[1]	**Bedeutung**
m	tc_name[8]	Name des Transaktionscodes (*tac_type*='A' oder 'D') oder der TAC-Queue (*tac_type*='Q'). Der Name darf bis zu 8 Zeichen lang sein. Zu Format und Eindeutigkeit des Namens siehe Abschnitt „Format und Eindeutigkeit der Objektnamen". Namen von zuvor gelöschten Objekten, die zu derselben Namensklasse gehören, dürfen nicht verwendet werden.
(m)	program[32]	Name des Teilprogramms, dem der Transaktionscode zugeordnet werden soll. Der Name kann bis zu 32 Zeichen lang sein. Das Teilprogramm muss bereits in der Konfiguration existieren oder vor dem Transaktionscode eingetragen werden. Für TAC-Queues ist dieser Parameter nicht erlaubt.
o	lock_code[4]	Lockcode (Zugriffsschutz), der dem Transaktionscode zugeordnet werden soll. Der Lockcode ist eine ganze Zahl. Er muss in dem Bereich liegen, der bei der KDCDEF-Generierung in MAX KEYVALUE definiert wurde. *lock_code* darf nicht zusammen mit *access_list* angegeben werden.
		Hinweis: Aufträge von einem Benutzer/Client werden nur bearbeitet, wenn sowohl das Keyset des Benutzers/Clients als auch das Keyset des LTERM-Partners, über den der Benutzer/Client mit der Anwendung verbunden ist, den Keycode enthalten, der dem Lockcode des Vorgangs-TACs entspricht.

	Feldname[1]	Bedeutung	
o	state	Legt fest, ob der Transaktionscode oder die TAC-Queue nach dem Erzeugen zunächst gesperrt sein soll oder nicht.	
		'Y'	Ein TAC wird nicht gesperrt (ON). Bei einer TAC-Queue ist Schreiben und Lesen erlaubt.
		'N'	Ein TAC wird gesperrt (OFF). Handelt es sich um den TAC eines KDCS-Teilprogramms vom *call_type*='B' oder 'N', dann ist der TAC als Vorgangs-TAC (1. TAC eines Vorgangs) gesperrt, aber nicht als Folge-TAC eines Vorgangs. Bei einer TAC-Queue ist Schreiben verboten und Lesen erlaubt.
		'H'	UTM nimmt keine Aufträge für den TAC an. Der TAC wird vollständig gesperrt (HALT). Wenn dieser TAC als Folge-TAC aufgerufen wird, dann wird der Vorgang mit PEND ER (74Z) beendet. Asynchron-Aufträge, die bereits in der Message Queue des TACs zwischengespeichert sind, werden nicht gestartet. Sie bleiben in der Message Queue, bis der Status des TACs wieder auf ON oder OFF gesetzt wird. Eine TAC-Queue ist für Schreib- und Lesezugriffe gesperrt.
		'K'	'K' darf nur für Asynchron-Transaktionscodes angegeben werden, die auch Vorgangs-TACs (*call_type*='B'oder 'F') sind, und für TAC-Queues. UTM nimmt Aufträge für den Transaktionscode an. Die Aufträge werden jedoch nicht bearbeitet, sondern lediglich in die Auftragswarteschlange des Transaktionscodes geschrieben. Sie werden bearbeitet, wenn Sie den Status des Transaktionscodes ändern in 'Y' oder 'N'. *state*='K' können Sie benutzen, um Aufträge zu sammeln, die erst zu einem Zeitpunkt, an dem die Anwendung weniger belastet ist (z.B. nachts), ausgeführt werden sollen. Um eine Überlastung des Pagepools durch zuviele zwischengespeicherte Aufträge zu vermeiden, sollten Sie die Auftragswarteschlange des Transaktionscodes durch den Parameter *qlev* begrenzen. Bei einer TAC-Queue ist Lesen verboten und Schreiben erlaubt.
		Für die Administrationskommandos KDCSHUT und KDCTAC setzt UTM immer *state*='Y', auch wenn Sie einen anderen Wert für *state* angeben. Auf diese Weise bleibt Ihre Anwendung immer administrierbar.	

	Feldname[1]	Bedeutung	
o	tacclass[2]	Darf nur angegeben werden, wenn bei der KDCDEF-Generierung mindestens eine TAC-Klasse erzeugt wurde. In *tacclass* geben Sie an, welcher TAC-Klasse der Transaktionscode zugeordnet werden soll. Folgendes ist zu beachten: • Ein Dialog-TAC *(tac_type*='D') kann nur einer TAC-Klasse zwischen 1 und 8 (1 <= *tacclass* <= 8) zugeordnet werden. • Ein Asynchron-TAC (*tac_type*='A') kann nur einer TAC-Klasse zwischen 9 und 16 (9 <= *tacclass* <= 16) zugeordnet werden. • Wenn Ihre Anwendung **ohne** TAC-PRIORITIES -Anweisung generiert ist, dann müssen alle Dialog-TACs (*tac_type*='D') von Teilprogrammen, die blockierende Aufrufe (z.B. den KDCS-Aufruf PGWT) verwenden, derselben Dialog-TAC-Klasse zugeordnet werden. Für diese TAC-Klasse muss PGWT=YES gesetzt sein. Entsprechend müssen alle Asynchron-TACs, die blockierende Aufrufe nutzen, der Asynchron-TAC-Klasse zugeordnet werden, für die PGWT=YES gesetzt ist. • Wenn Ihre Anwendung **mit** TAC-PRIORITIES -Anweisung generiert ist, dann können Dialog-TACs von Teilprogrammen, die blockierende Aufrufe durchlaufen, jeder beliebigen Dialog-TAC-Klasse zugeordnet werden. Sie müssen dann lediglich *pgwt*='Y'setzen. Analoges gilt für Asynchron-TACs.	
		Standard (vorausgesetzt es existiert mindestens eine TAC-Klasse): Dialog-TACs werden keiner TAC-Klasse zugeordnet, Asynchron-TACs werden der TAC-Klasse 16 zugeordnet.	
o	admin	Gibt an, welche Berechtigung ein Benutzer oder Client haben muss, um diesen Transaktionscode bzw. einen Vorgang, der diesen Transaktionscode als Folge-TAC enthält, aufrufen zu können. Bei einer TAC-Queue bezieht sich die Berechtigung auf schreibende und lesende Zugriffe. Mögliche Werte sind:	
		'Y'	Diesen Transaktionscode darf nur ein Benutzer mit Administrationsberechtigung aufrufen. *admin*='Y' muss den Transaktionscodes von Administrationsprogrammen zugeordnet werden, die nicht nur lesend auf Anwendungsdaten zugreifen. Bei einer TAC-Queue darf nur ein Benutzer mit Administrationsberechtigung Nachrichten aus dieser Queue lesen bzw. in diese Queue schreiben.
		'N'	Es ist keine Administrationsberechtigung zum Aufrufen des Transaktionscodes bzw. für den Zugriff auf die TAC-Queue erforderlich. Teilprogramme, die über einen Transaktionscode mit *admin*='N' gestartet werden, dürfen keine KDCADMI-Aufrufe absetzen.
		'R'	Wie bei *admin*='N' ist keine Administrationsberechtigung erforderlich, um diesen Transaktionscode aufzurufen bzw. auf die TAC-Queue zuzugreifen. Das zugehörige Teilprogramm darf jedoch alle Funktionen von KDCADMI nutzen, die lesend auf die Anwendungsdaten zugreifen.

	Feldname[1]	**Bedeutung**	
o	call_type	Legt fest, ob mit dem Transaktionscode ein Service gestartet wird oder ob er Folge-TAC in einem Vorgang ist. Folgende Angaben sind möglich:	
		'B'	Der TAC kann sowohl 1. TAC als auch Folge-TAC in einem Vorgang sein (BOTH).
		'F'	Der TAC kann nur der 1. TAC in einem Vorgang sein (FIRST).
		'N'	Der TAC kann nur Folge-TAC in einem Vorgang sein (NEXT).
o	exit_name[32]	Name des Event-Exit VORGANG, der diesem TAC zugeordnet werden soll. *exit_name* darf nur angegeben werden, wenn *call_type*='F' oder 'B' gesetzt wird. Der in *exit_name* angegebene VORGANG-Exit muss bereits als Teilprogramm der Anwendung in der Konfiguration enthalten sein (dynamisch als Objekttyp KC_PROGRAM oder mit der KDCDEF-Anweisung PROGRAM). Ist das Teilprogramm in *program* in ein Lademodul mit Lademodus ONCALL eingebunden, dann muss der VORGANG-Exit in demselben Lademodul enthalten sein.	
o	qlev[5]	Ist nur relevant bei Asynchron-TACs (*tac_type*='A') oder TAC-Queues (*tac_type*='Q'). In *qlev* geben Sie an, wieviele Asynchron-Nachrichten maximal gleichzeitig in der Message Queue des Transaktionscodes bzw. Nachrichten in der TAC-Queue zwischengespeichert werden sollen. UTM berücksichtigt die Aufträge erst am Ende der Transaktion. Daher kann die in *qlev* festgelegte Anzahl von Nachrichten für eine Message Queue überschritten werden, wenn in einer Transaktion mehrere Nachrichten für dieselbe Queue erzeugt wurden. Wird die in *qlev* festgelegte Anzahl überschritten, so ist das Verhalten von UTM abhängig von der Einstellung bei *q_mode*.	
		Minimalwert: '0', Maximalwert: '32767'	
		Wird in *qlev* ein Wert > 32767 angegeben, dann setzt UTM ohne Meldung den Standardwert.	
o	tac_type	Legt fest, ob Aufträge an diesen Transaktionscode im Dialog oder asynchron bearbeitet werden oder ob eine TAC-Queue erzeugt wird:	
		'D'	Der Transaktionscode ist ein Dialog-TAC
		'A'	Der Transaktionscode ist ein Asynchron-TAC
		'Q'	Es wird eine TAC-Queue erzeugt. In eine solche Queue kann mit einem DPUT-Aufruf eine Nachricht geschrieben und aus der Queue kann mit einem DGET-Aufruf gelesen werden.
o	real_time_sec[5]	Legt die Realzeit in Sekunden fest, die ein Teilprogrammlauf, der mit diesem TAC gestartet wurde, maximal verbrauchen darf. Läuft das Teilprogramm länger, bricht UTM den Vorgang ab. *real_time_sec*='0' bedeutet keine Einschränkung der Realzeit.	
		Minimalwert: '0', Maximalwert: '32767'	

	Feldname[1]	**Bedeutung**
o	cpu_time_msec[8]	*Nur auf BS2000-Systemen:* Legt die CPU-Zeit in Millisekunden fest, die ein Teilprogrammlauf, der mit diesem TAC gestartet wurde, maximal verbrauchen darf. Läuft das Teilprogramm länger, bricht UTM den Vorgang ab. *cpu_time_msec*='0' bedeutet keine Einschränkung der CPU-Zeit.
		Minimalwert: '0', Maximalwert: '86400000'
		Die Werte 1 bis 999 sind unzulässig und werden von UTM auf den Wert 1000 abgebildet.
o	dbkey[8]	*Nur auf BS2000-Systemen:* Ist nur relevant, wenn das zum Transaktionscode gehörende Teilprogramm Datenbankaufrufe absetzt und das Datenbanksystem mit UTM gekoppelt ist.
		In *dbkey* geben Sie den Datenbankschlüssel an, den UTM bei einem Datenbankaufruf aus dem Teilprogramm an das Datenbanksystem übergibt. Das Format des Schlüssels ist abhängig vom verwendeten Datenbanksystem. Der Schlüssel ist maximal 8 Zeichen lang. Zur Zeit wird *dbkey* nur für UDS unterstützt.
		Der Wert *dbkey*='UTM' bewirkt, dass der Wert des Startparameters DBKEY an die Datenbank übergeben wird (siehe openUTM-Handbuch „Einsatz von UTM-Anwendungen auf BS2000-Systemen“; Startparameter).
o	runprio[3]	*Nur auf BS2000-Systemen:* Run-Priorität des Prozesses im Betriebssystem, in dem das zum Transaktionscode gehörende Teilprogramm abläuft. *runprio*='0' bedeutet, dass dem Transaktionscode keine spezielle Run-Priorität zugeordnet wird.
		Minimalwert: '30' (höchste Priorität), Maximalwert: '255' (niedrigste Priorität)
o	api	UTM-Programmschnittstelle, die das zu dem Transaktionscode gehörende Teilprogramm verwendet.
		'K': KDCS 'X': X/Open-Schnittstelle XATMI 'C': X/Open-Schnittstelle CPI-C

	Feldname[1]	Bedeutung	
o	satadm	*Nur auf BS2000-Systemen:* Legt fest, ob zum Aufruf des Transaktionscodes die UTM-SAT-Administrationsberechtigung erforderlich ist.	
		'Y'	Der Transaktionscode darf nur von Benutzern und Partner-Anwendungen aufgerufen werden, die die UTM-SAT-Administrationsberechtigung haben. *satadm*='Y' muss gesetzt werden, wenn der Transaktionscode die UTM-SAT-Administrationsfunktionen nutzt.
		'N'	Die UTM-SAT-Administrationsberechtigung ist für den Aufruf des Transaktionscodes nicht erforderlich.
o	satsel	*Nur auf BS2000-Systemen:* Art der SAT-Protokollierung für diesen Transaktionscode.	
		'B'	Es sollen erfolgreiche und nicht-erfolgreiche Ereignisse protokolliert werden (BOTH).
		'S'	Es sollen nur erfolgreiche Ereignisse protokolliert werden (SUCC).
		'F'	Es sollen nur nicht-erfolgreiche Ereignisse protokolliert werden (FAIL).
		'N'	Es wird keine TAC-spezifische SAT-Protokollierung definiert.
		Voraussetzung für die Protokollierung ist, dass die SAT-Protokollierung für die Anwendung eingeschaltet ist (zur SAT-Protokollierung siehe openUTM-Handbuch „Anwendungen generieren“)	
o	tacunit[4]	Ist nur relevant, wenn die Anwendung Accounting-Funktionen nutzt (siehe openUTM-Handbuch „Anwendungen generieren“, KDCDEF-Steueranweisung ACCOUNT und openUTM-Handbuch „Einsatz von UTM-Anwendungen“ ,SAT-Protokollierung).	
		In *tacunit* geben Sie die Zahl der Verrechnungseinheiten an, die in der Abrechnungsphase jedem Benutzer für den Aufruf dieses Transaktionscodes berechnet wird. Für *tacunit* sind nur ganzzahlige Angaben erlaubt.	
		Minimalwert: '0', Maximalwert: '4095''	
o	pgwt	dürfen Sie nur angeben, wenn in Ihrer Anwendung die Aufträge an TAC-Klassen prioritätengesteuert bearbeitet werden, d.h. die KDCDEF-Generierung enthält die Anweisung TAC-PRIORITIES.	
		In *pgwt* geben Sie an, ob in einem Teilprogrammlauf, der für diesen Transaktionscode gestartet wird, blockierende Aufrufe (z.B. PGWT) durchgeführt werden dürfen.	
		'Y'	Blockierende Aufrufe dürfen durchgeführt werden. 'Y' darf nur angegeben werden, wenn Sie den TAC einer TAC-Klasse zuordnen.
		'N'	Blockierende Aufrufe dürfen nicht durchgeführt werden.

	Feldname[1]	**Bedeutung**	
o	encryption_level	ist nur für Vorgangs-TACs (*call_type*='F' oder 'B') erlaubt. In *encryption_level* legen Sie fest, ob Nachrichten für diesen Transaktionscode verschlüsselt werden müssen oder nicht.	
		'2'	(Level 2) Für den Zugriff auf den Transaktionscode ist die Verschlüsselung der Eingabe-Nachrichten nach dem AES-CBC Algorithmus erforderlich.
		'5'	(Level 5) nur auf Unix-, Linux- und Windows-Systemen Für den Zugriff auf den Transaktionscode ist die Verschlüsselung der Eingabe-Nachrichten nach dem AES-GCM Algorithmus erforderlich.
			Falls *encryption_level* ='2' oder '5' angegeben wird, dann kann der Client über diesen Transaktionscode nur dann einen Vorgang (Service) starten, wenn er eine der folgenden Bedingungen erfüllt: • Er ist als „trusted“ Client generiert (siehe *kc_pterm_str* oder *kc_tpool_str* Feld *encryption_level*). • Er hat die Eingabe-Nachricht mit mindestens der angegebenen Verschlüsselungsebene verschlüsselt an den Transaktionscode übertragen. Verschlüsselt ein „not-trusted“ Client die Eingabe-Nachricht nicht oder nicht ausreichend, so wird kein Vorgang gestartet.
			Wird der Transaktionscode ohne Benutzerdaten aufgerufen oder durch Vorgangskettung gestartet, dann muss der Client die Fähigkeit zur Verschlüsselung haben, da UTM alle Dialog-Ausgabe-Nachrichten verschlüsselt überträgt, und, bei Mehrschritt-Vorgängen, alle weiteren Eingabe-Nachrichten vom „not-trusted“ Client verschlüsselt erwartet.
		'N'	(NONE) Eine Nachrichtenverschlüsselung ist nicht erforderlich.
o	access_list[8]	Hiermit bestimmen Sie ein Keyset, das die Zugriffsrechte von Benutzern für diesen Transaktionscode regelt. Das Keyset muss zuvor dynamisch erzeugt oder bereits bei der Generierung definiert worden sein. *access_list* darf nicht zusammen mit *lock_code* angegeben werden. Ein Benutzer kann nur dann auf den Transaktionscode zugreifen, wenn das Keyset des Benutzers, das Keyset des LTERM-Partners, über den der Benutzer angemeldet ist, und das angegebene Keyset mindestens einen gemeinsamen Keycode enthalten. Geben Sie weder *access_list* noch *lock_code* an, dann ist der Transaktionscode nicht geschützt und jeder Benutzer kann den Transaktionscode aufrufen.	

	Feldname[1]	Bedeutung	
o	q_mode	legt fest, wie sich UTM verhält, wenn die maximale Anzahl der noch nicht ausgeführten gesicherten Aufträge an diesen Asynchron-TAC bzw. an die TAC-Queue erreicht ist. Mögliche Werte sind:	
		'S'	UTM lehnt weitere Aufträge ab.
		'W'	Nur bei *tac_type*='Q': UTM nimmt weitere Nachrichten auf, löscht aber die ältesten in der Queue stehenden Nachrichten.
o	q_read_acl[8]	Nur bei *tac_type*='Q': legt die Rechte fest (Name eines Keysets), die ein Benutzer benötigt, um Nachrichten aus dieser Queue lesen und löschen zu können. Ein Benutzer kann nur dann lesend auf diese TAC-Queue zugreifen, wenn das Keyset des Benutzers und das Keyset des logischen Terminals, über das der Benutzer angemeldet ist, jeweils mindestens einen Keycode enthalten, der auch in dem angegebenen Keyset enthalten ist. Enthält *q_read_acl* keinen Wert, dann können alle Benutzer Nachrichten aus dieser Queue lesen und dabei löschen.	
o	q_write_acl[8]	Nur bei *tac_type*='Q': legt die Rechte fest (Name eines Keysets), die ein Benutzer benötigt, um Nachrichten in diese Queue zu schreiben. Ein Benutzer kann nur dann schreibend auf diese TAC-Queue zugreifen, wenn das Keyset des Benutzers und das Keyset des logischen Terminals, über das der Benutzer angemeldet ist, jeweils mindestens einen Keycode enthalten, der auch in dem angegebenen Keyset enthalten ist. Enthält *q_write_acl* keinen Wert, dann können alle Benutzer Nachrichten in diese Queue schreiben.	
o	dead_letter_q	legt fest, ob eine Asynchron-Nachricht in der Dead Letter Queue aufbewahrt werden soll, wenn sie nicht ordnungsgemäß verarbeitet wurde und keine Redelivery erfolgt ist.	
		'Y'	Nachrichten an diesen Asynchron-TAC oder diese TAC-Queue, die nicht verarbeitet werden konnten, werden in der Dead Letter Queue gesichert, sofern sie nicht erneut zugestellt werden (Redelivery) und (bei Message-Komplexen) kein negativer Quittungs-Auftrag definiert wurde.
		'N'	Nachrichten an diesen Asynchron-TAC oder diese TAC-Queue, die nicht verarbeitet werden konnten, werden nicht in der Dead Letter Queue gespeichert. Dieser Wert muss für alle Dialog-TACs und für Asynchron-TACs mit CALL=NEXT, sowie für KDCMSGTC und KDCDLETQ angegeben werden.

[1] Alle nicht aufgeführten Felder der Datenstruktur *kc_tac_str* und alle für das verwendete Betriebssystem nicht relevanten Felder sind mit binär null zu versorgen. Die Datenstruktur ist im Abschnitt "kc_tac_str - Transaktionscodes lokaler Services" vollständig beschrieben.

11.2.3.9 obj_type = KC_USER

Um eine neue Benutzerkennung zu erzeugen, müssen Sie die Datenstruktur *kc_user_str* über den Datenbereich legen.

Jeder Benutzerkennung steht automatisch eine permanente Queue zu Verfügung, die mit dem Namen der Benutzerkennung angesprochen wird. Der Zugriff von fremden Benutzern auf diese USER-Queue wird mit den Werten in den Feldern *q_read_acl* und *q_write_acl* gesteuert. Die maximale Anzahl von Nachrichten, die zwischengespeichert werden können, und das Verhalten von UTM, wenn dieser Wert erreicht ist, wird von den Werten in den Feldern *qlev* und *q_mode* bestimmt.

Der folgenden Tabelle können Sie entnehmen, wie die Felder der Datenstruktur zu versorgen sind.

	Feldname [1]	**Bedeutung**
m	us_name[8]	Name der Benutzerkennung. Er kann bis zu 8 Zeichen lang sein. Stimmt der Name der Benutzerkennung mit dem Namen eines LTERM-Partners überein, der einem UPIC-Client oder einer TS- Anwendung, aber keiner Benutzerkennung zugeordnet ist, dann kann sich kein Benutzer unter dieser Benutzerkennung an die UTM- Anwendung anmelden. UTM ordnet diese Benutzerkennung dann exklusiv dem Client zu. Zum Format des Namens und zur Eindeutigkeit siehe im Abschnitt „Format und Eindeutigkeit der Objektnamen" . Namen von zuvor „verzögert“ gelöschten Objekten (mit KC_DELAY) derselben Namensklasse dürfen nicht verwendet werden.
o	kset[8]	Keyset der Benutzerkennung. Das Keyset muss zuvor dynamisch erzeugt oder statisch generiert worden sein. Das Keyset legt die Zugriffsrechte des Benutzers /Clients fest, der sich unter dieser Benutzerkennung bei der Anwendung anmeldet.
o	state	Legt fest, ob die Benutzerkennung zunächst gesperrt sein soll oder nicht. Mit einer gesperrten Benutzerkennung kann sich kein Benutzer/Client bei der Anwendung anmelden. Sie muss vom Administrator explizit freigegeben werden.
		'Y': Die Benutzerkennung soll nicht gesperrt werden (ON). 'N': Die Benutzerkennung soll gesperrt werden (OFF).

	Feldname [1]	**Bedeutung**
o	card_position[3] card_string_lth[3] card_string_type card_string[200]	*Nur auf BS2000-Systemen:* Diese Felder sind nur relevant, wenn für diese Benutzerkennung der Zugang zur Anwendung nur mit einer Magnetstreifenkarte möglich sein soll. Die Felder geben an, welches Teilfeld der Ausweisinformation auf der Magnetstreifenkarte geprüft wird und welche Informationen in diesem Teilfeld stehen müssen. Die Angabe von *card...* schließt die Angabe von *principal* aus.
		In den einzelnen Feldern müssen Sie Folgendes angeben:
		card_position Nummer des Byte auf der Magnetstreifenkarte, mit dem die zu prüfende Information beginnt. *card_string_lth* Länge der zu prüfenden Ausweisinformation in Byte. Maximalwert: '100', Minimalwert: '1'
		card_position und *card_string_lth* müssen ein Teilfeld der Ausweisinformation definieren, das in dem Bereich liegt, der durch den Generierungsparameter MAX CARDLTH definiert ist.
		card_string_type Codierung der zu prüfenden Ausweisinformation:
		'X' — Die Ausweisinformation wird als hexadezimaler String übergeben.
		'C' — Die Ausweisinformation wird als Character-String übergeben.
		card_string Zeichenfolge, die im zu prüfenden Teilbereich auf der Magnetstreifenkarte stehen muss. Der Inhalt dieses Feldes ist bei card_string_type='C' nur in der Länge relevant, die in card_string_lth angegeben wird. Bei card_string_type='X' ist der Inhalt nur in der Länge 2 card_string_lth relevant.
		Für die Übergabe der Ausweisinformation steht die Union *kc_string* zur Verfügung (siehe im Abschnitt "kc_user_str, kc_user_fix_str, kc_user_dyn1_str bzw. kc_user_dyn2_str - Benutzerkennungen").

	Feldname [1]	**Bedeutung**
o	password16	Passwort für diese Benutzerkennung. Das Passwort darf maximal 16 Zeichen lang sein. Das angegebene Passwort muss der in *protect_pw_compl* und *protect_pw16_lth* angegebenen Komplexitätsstufe entsprechen. Zusätzlich müssen Sie im Feld *password_type* angeben, wie UTM die Angabe in *password* interpretieren soll. Das Passwort muss aus Zeichen bestehen, die in der UTM- Anwendung erlaubt sind, siehe openUTM-Handbuch „Anwendungen generieren“, USER-Anweisung. Auf BS2000-Systemen schließt die Angabe von *password16* die Angabe von *principal* aus.
		Für die Übergabe des Passworts steht Ihnen die Union *kc_pw16* zur Verfügung.

		`union kc_pw16`
		`char x[32]; /* for X'...' */`
		`char c[16]; /* for C'...' */`

		In UTM-Anwendungen auf BS2000-Systemen können Sie das Passwort entweder als Character-String oder als hexadezimale Zeichenfolge angeben. Bei einem hexadezimalen Passwort (*password_type* ='X') wird jedes Halbbyte als ein Zeichen dargestellt. Geben Sie ein Passwort an, das aus weniger als 16 Zeichen besteht, dann muss *password16* rechts mit Leerzeichen (*password_type* ='C'), bzw. mit dem hexadezimalen Wert für Leerzeichen (*password_type* ='X') aufgefüllt werden.
		In UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen müssen Sie das Passwort immer als Character-String übergeben (Feld *password16.c*). Geben Sie ein Passwort ein, das aus weniger als 16 Zeichen besteht, dann muss *password16.c* rechts mit Leerzeichen aufgefüllt werden.
		password16 müssen Sie angeben, wenn *password_type* ='C' oder 'X' ist. *password16* dürfen Sie nicht angeben, wenn *password_type* = 'R' oder 'N' ist.
		Soll eine Benutzerkennung ohne Passwort erzeugt werden, dann dürfen Sie in *password16* und in *password_type* nichts angeben. Für *protect_pw_compl* muss dann '0' und für *protect_pw16_lth* '00' gesetzt werden (Standard).

	Feldname [1]	**Bedeutung**	
o	password_type	In *password_type* müssen Sie angeben, wie das Passwort in *password* zu interpretieren ist. Folgende Angaben sind möglich:	
		'C'	Das Passwort in *password* ist als Character-String zu interpretieren.
		'X'	Das Passwort in *password* ist als hexadezimales Passwort zu interpretieren. Nur für Benutzerkennungen einer UTM- Anwendung auf einem BS2000-System erlaubt.
		'N'	Kein Passwort. Im Feld *password* darf nichts angegeben werden
		'R'	Als Passwort wird ein zufälliges Passwort erzeugt (random). Bevor sich der so generierte Benutzer anmelden kann, muss der Administrator das Passwort explizit neu setzen.
o	password_dark	Legt fest, ob das Passwort am Terminal dunkelgesteuert eingegeben werden muss.	
		'Y'	UTM fordert den Benutzer nach dem KDCSIGN in einem Zwischen-Dialog auf, das Passwort in ein dunkelgesteuertes Feld einzugeben.
		'N'	Der Benutzer übergibt das Passwort direkt beim KDCSIGN. Das Passwort ist bei der Anmeldung am Bildschirm sichtbar (Standard).
		Sie können *password_dark* ='Y' auch setzen, wenn Sie kein Passwort angeben. Wird der Benutzerkennung dann zu einem späteren Zeitpunkt ein Passwort zugeordnet (z.B. mit KC_MODIFY_ OBJECT), ist es dunkelgesteuert.	
		Hinweis : Bei Anwendungen auf Unix-, Linux- und Windows-Systemen werden die Passwörter grundsätzlich nicht dunkelgesteuert.	

	Feldname [1]	**Bedeutung**
o	format_attr format_name[7]	*Nur auf BS2000-Systemen:* Mit Hilfe dieser Felder können Sie der Benutzerkennung ein Benutzer-spezifisches Startformat zuweisen. Sie müssen *format_name* und *format_attr* angeben.
		Voraussetzung für das Zuweisen eines Startformats ist, dass ein Formatierungssystem generiert ist (KDCDEF-Anweisung FORMSYS). Ist das Startformat ein #Format, dann muss zusätzlich ein Anmelde-Vorgang generiert sein.
		In *format_attr* geben Sie das Formatkennzeichen des Startformats an:
		'A' für das Formatattribut ATTR (+Format). 'N' für das Formatattribut NOATTR (*Format). 'E' für das Formatattribut EXTEND (#Format).
		Bedeutung der Formatattribute siehe im Abschnitt "kc_user_str, kc_user_fix_str, kc_user_dyn1_str bzw. kc_user_dyn2_str - Benutzerkennungen" (format_attr, format_name).
		In *format_name* geben Sie den Namen des Startformats an. Der Name kann bis zu 7 Zeichen lang sein und darf nur alphanumerische Zeichen enthalten.
o	locale_lang_id[2] locale_terr_id[2] locale_ccsname[8]	*Nur auf BS2000-Systemen:* Sprachumgebung (Locale) der Benutzerkennung. Die Sprachumgebung ist relevant, wenn Nachrichten und Meldungen der Anwendung in verschiedenen Sprachen ausgegeben werden. Zur Mehrsprachigkeit siehe openUTM-Handbuch „Anwendungen generieren“.
		In *locale_lang_id* geben Sie das Sprachkennzeichen der Landessprache an, in der Meldungen und Nachrichten übergeben werden sollen. Es ist maximal 2 Byte lang.
		In *locale_terr_id* geben Sie das Territorialkennzeichen an. Es kennzeichnet territoriale Besonderheiten in der Landessprache. Es ist maximal 2 Byte lang.
		In *locale_ccsname* geben Sie den CCS-Namen des erweiterten Zeichensatzes an (**c** oded **c** haracter **s** et), der für die Ausgabe verwendet werden soll. Der CCS-Name ist bis zu 8 Byte lang. Er muss zu einem auf dem BS2000-System definierten EBCDIC-Zeichensatz gehören (siehe auch Benutzerhandbuch zu XHCS).

	Feldname [1]	**Bedeutung**	
o	protect_pw16_lth	Legt fest, wie viele Zeichen das Passwort der Benutzerkennung mindestens haben muss, damit es von UTM akzeptiert wird (minimale Länge des Passworts). Das Passwort einer Benutzerkennung kann nur gelöscht werden, wenn *protect_pw16_lth* ='00' ist.	
		Maximalwert: '16', Der Minimalwert ist abhängig von der in *protect_pw_compl* angegebenen Komplexitätsstufe. Der Minimalwert von *protect_pw16_lth* ist: '0' bei *protect_pw_compl* ='0' '1' bei *protect_pw_compl* ='1' '2' bei *protect_pw_compl* ='2' '3' bei *protect_pw_compl* ='3'	
o	protect_pw_compl	Legt die Komplexitätsstufe fest, der das Passwort der Benutzerkennung genügen muss.	
		'0'	(NONE) Es kann jede beliebige Zeichenfolge als Passwort angegeben werden.
		'1'	(MIN) Im Passwort dürfen maximal zwei aufeinanderfolgende Zeichen gleich sein. Die minimale Länge des Passworts ist ein Zeichen.
		'2'	(MEDIUM) In dem Passwort dürfen maximal zwei aufeinanderfolgende Zeichen gleich sein. Das Passwort muss mindestens einen Buchstaben und eine Ziffer enthalten. Das Passwort muss mindestens 2 Zeichen lang sein.
		'3'	(MAX) Passwort dürfen maximal zwei aufeinanderfolgende Zeichen gleich sein. Das Passwort muss mindestens einen Buchstaben, eine Ziffer und ein Sonderzeichen enthalten. Die minimale Länge ist 3 Zeichen. Sonderzeichen sind alle Zeichen, die von a-z, A-Z, 0-9 und Leerzeichen verschieden sind.

	Feldname [1]	**Bedeutung**	
o	protect_pw_time[3]	Legt fest, wieviele Tage das Passwort maximal gültig ist (Gültigkeitsdauer). Wird *protect_pw_time*='0' angegeben, so ist die Gültigkeitsdauer nicht beschränkt.	
		Minimalwert: '0', Maximalwert: '180'	
o	restart	Legt fest, ob UTM Vorgangsdaten für die Benutzerkennung sichert, damit beim nächsten Anmelden unter dieser Benutzerkennung ein Vorgangswiederanlauf möglich ist.	
		'Y':UTM sichert Vorgangsdaten. 'N': UTM sichert keine Vorgangsdaten.	
o	permit	Legt die Administrationsberechtigung der Benutzerkennung fest.	
		'A'	(ADMIN) Die Benutzerkennung soll Administrationsfunktionen in der lokalen Anwendung ausführen können.
		'N'	(NONE) Die Benutzerkennung soll keine Administrationsberechtigung haben. In UTM-Anwendungen auf BS2000-Systemen dürfen unter dieser Benutzerkennung auch keine UTM-SAT-Administrationsfunktionen ausgeführt werden.
		'B'	(BOTH) Nur auf BS2000-Systemen: Unter der Benutzerkennung dürfen Administrations- und UTM-SAT-Administrationsfunktionen ausgeführt werden.
		'S'	(SAT) Nur auf BS2000-Systemen: Die Benutzerkennung hat UTM- SAT-Administrationsberechtigung. Es dürfen Preselection- Funktionen ausgeführt werden.
o	satsel	*Nur auf BS2000-Systemen:* Legt die Art der SAT-Protokollierung für die Benutzerkennung fest.	
		'B'	Es sollen erfolgreiche und nicht erfolgreiche Ereignisse protokolliert werden (BOTH).
		'S'	Es sollen nur erfolgreiche Ereignisse protokolliert werden (SUCC).
		'F'	Es sollen nur nicht erfolgreiche Ereignisse protokolliert werden (FAIL).
		'N'	Es wird keine Benutzer-spezifische SAT-Protokollierung definiert (NONE).
		Voraussetzung für die Protokollierung ist, dass die SAT-Protokollierung für die Anwendung eingeschaltet ist (zur SAT-Protokollierung siehe openUTM-Handbuch „Anwendungen generieren“ und openUTM-Handbuch „Einsatz von UTM-Anwendungen auf BS2000-Systemen“)	

	Feldname [1]	**Bedeutung**
o	protect_pw_min_time[3]	Legt die minimale Gültigkeitsdauer des Passworts in Tagen fest.
		Nach der Änderung des Passworts darf der Benutzer das Passwort frühestens nach Ablauf der minimalen Gültigkeitsdauer erneut ändern. Wird eine minimale Gültigkeitsdauer von einem Tag angegeben, so darf das Passwort frühestens um 0.00 Uhr des folgenden Tages geändert werden (Ortszeit der Generierung).
		Nach einer Passwortänderung durch den Administrator bzw. nach einer Neugenerierung kann der Benutzer das Passwort immer ändern, unabhängig davon, ob die minimale Gültigkeitsdauer abgelaufen ist oder nicht.
		protect_pw_min_time darf nicht größer als *protect_pw_time* (maximale Gültigkeitsdauer) sein.
		Minimum: '0' Maximum: '180'
o	qlev[5]	legt fest, wieviele Nachrichten in der Nachrichten-Queue des Benutzers maximal zwischengespeichert werden können. Wird der Schwellwert überschritten, dann hängt das Verhalten vom Wert im Feld *q_mode* ab. Bei *qlev* =0 können keine Nachrichten in der Queue gespeichert werden, bei *qlev* =32767 ist die Queue-Länge nicht begrenzt. Minimalwert: 0, Maximalwert: 32767
o	q_read_acl[8]	legt die Rechte fest (Name eines Keysets), die ein fremder Benutzer benötigt, um Nachrichten aus dieser USER-Queue lesen und löschen zu können. Ein fremder Benutzer kann nur dann lesend auf diese USER-Queue zugreifen, wenn das Keyset seiner Benutzerkennung und das Keyset des logischen Terminals, über das der Benutzer angemeldet ist, jeweils mindestens einen Keycode enthalten, der auch in dem angegebenen Keyset enthalten ist. Enthält *q_read_acl* keinen Wert, dann können alle Benutzer Nachrichten aus dieser Queue lesen und dabei löschen.
o	q_write_acl[8]	legt die Rechte fest (Name eines Keysets), die ein fremder Benutzer benötigt, um Nachrichten in diese USER-Queue zu schreiben. Ein fremder Benutzer kann nur dann schreibend auf diese Queue zugreifen, wenn das Keyset der Benutzerkennung und das Keyset des logischen Terminals, über das der Benutzer angemeldet ist, jeweils mindestens einen Keycode enthalten, der auch in dem angegebenen Keyset enthalten ist. Enthält *q_write_acl* keinen Wert, dann können alle Benutzer Nachrichten in diese Queue schreiben.

	Feldname [1]	Bedeutung	
o	q_mode	legt fest, wie sich UTM verhält, wenn die maximale Anzahl der noch nicht ausgeführten Aufträge in der Queue des Benutzers erreicht ist. Mögliche Werte sind:	
		'S'	UTM lehnt weitere Aufträge ab (Standard).
		'W'	UTM nimmt weitere Nachrichten auf, löscht aber die ältesten in der Queue stehenden Nachrichten.
o	principal[100]	*Nur auf BS2000-Systemen:* legt fest, dass die Authentisierung des Benutzers über Kerberos erfolgen soll. Die Angabe von *principal* schließt die Angabe von *card* und *password* aus. *principal* muss als alphanumerische Zeichenfolge der Form `windowsaccount@NT-DNS-REALM-NAME` angegeben werden. `windowsaccount` : Domänen-Kennung des Benutzers `NT-DNS-REALM-NAME` : DNS-Name der Active-Directory-Domäne	

[1] Alle nicht aufgeführten Felder der Datenstruktur *kc_user_str* und alle für das verwendete Betriebssystem nicht relevanten Felder sind mit binär null zu versorgen. Die Datenstruktur ist im Abschnitt "kc_user_str, kc_user_fix_str, kc_user_dyn1_str bzw. kc_user_dyn2_str - Benutzerkennungen" vollständig beschrieben.

11.2.3.10 Returncodes

Neben den im Abschnitt „Returncodes" aufgelisteten Returncodes können zusätzlich folgende Returncodes auftreten. Einige dieser Returncodes können unabhängig vom angegebenen Objekttyp auftreten, andere treten nur für bestimmte Objekttypen auf.

Maincode = KC_MC_DATA_INVALID Ein Feld der Datenstruktur im Datenbereich enthält einen ungültigen Wert. **Subcodes:**
KC_SC_NOT_NULL In der Datenstruktur ist ein Feld, das mit binär null belegt sein muss, anders versorgt.
KC_SC_NO_INFO In der Datenstruktur ist ein Feld mit einem ungültigen Wert besetzt.

Maincode = KC_MC_REJECTED Der Aufruf wurde von UTM abgewiesen. **Subcodes:**
KC_SC_NAME_MISSING Für das Objekt, das konfiguriert werden soll, wurde kein Name angegeben.
KC_SC_TAB_FULL Es können keine Objekte des angegebenen Objekttyps mehr erzeugt werden, da die bei der KDCDEF-Generierung reservierten Tabellenplätze bereits belegt sind bzw. keine Tabellenplätze für diesen Objekttyp reserviert wurden. Es ist zu beachten, dass Tabellenplätze von verzögert gelöschten Objekten nicht freigegeben werden.
KC_SC_EXISTENT Es existiert bereits ein Objekt mit dem angegebenen Objektnamen, das zu derselben Namensklasse gehört (siehe Abschnitt „Format und Eindeutigkeit der Objektnamen"). Es ist zu beachten, dass Namen von verzögert (mit KC_DELAY) gelöschten Objekten nicht wieder vergeben werden können.
KC_SC_OBJ_DEL Das Objekt, das konfiguriert werden soll, wurde verzögert gelöscht.
KC_SC_INVALID_NAME Der Objektname fängt mit 'KDC' an.
KC_SC_NO_GLOB_CHANG_POSSIBLE Nur bei UTM-Cluster-Anwendungen: Keine globalen Administrationsänderungen möglich, da die Generierung der Knoten-Anwendungen zur Zeit nicht konsistent ist.

KC_SC_GLOB_CRE_DEL_LOCKED

Nur bei UTM-Cluster-Anwendungen:
Das Erzeugen eines Objekts ist zur Zeit nicht möglich, weil in einer Knoten-Anwendung das Erzeugen oder Löschen eines Objekts oder das Erzeugen, Löschen oder Aktivieren eines RSA-Schlüsselpaars noch nicht abgeschlossen ist.

KC_SC_JCTL_RT_CODE_NOT_OK

Nur bei UTM-Cluster-Anwendungen:
UTM-interner Fehler.
Bitte wenden Sie sich an die Systembetreuung.

Maincode = KC_MC_REJECTED_CURR

Der Aufruf kann zur Zeit nicht bearbeitet werden.

Subcode:

KC_SC_INVDEF_RUNNING

Es läuft gerade ein inverser KDCDEF bzw. es ist gerade ein inverser KDCDEF-Lauf in Vorbereitung (asynchron), siehe KC_CREATE_STATEMENTS im Abschnitt "KC_CREATE_STATEMENTS - KDCDEF-Steueranweisungen erzeugen (inverser KDCDEF)".

Maincode = KC_MC_RECBUF_FULL

Der Puffer mit Wiederanlauf-Informationen ist voll. Die Größe des Puffers wird mit der KDCDEF-Steueranweisung MAX, Operand RECBUF festgelegt.
Siehe openUTM-Handbuch „Anwendungen generieren“.

Subcode:

KC_SC_NO_INFO

Returncodes bei obj_type = KC_CON:

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen.

Subcodes:

KC_SC_PROCESSOR_MISSING (nur auf BS2000-Systemen)

In *pronam_long* wurde kein Rechnername angegeben. Die Angabe von *pronam_long* ist in UTM-Anwendungen auf BS2000-Systemen Pflicht.

KC_SC_PROCESSOR_NOT_ALLOWED

In *pronam_long* wurde ein Rechnername länger als 8 Zeichen angegeben, der keinen Punkt ('.') enthält und somit kein DNS-Name sein kann.

KC_SC_LPAP_MISSING

Es wurde kein LPAP-Partner angegeben.

KC_SC_LPAP_NOT_EXISTENT

Der angegebene LPAP-Partner existiert nicht.

KC_SC_BCAMAPPL_NOT_EXISTENT

Der in *bcamappl* angegebene Anwendungsname existiert nicht.

KC_SC_TPROT_NOT_ALLOWED (nur auf Unix-, Linux- und Windows-Systemen)

Es wird ein BCAMAPPL mit *t_prot=socket* referenziert.

KC_SC_INVALID_LISTENID (nur auf Unix-, Linux- und Windows-Systemen)

Die in *listener_port* angegebene Nummer ist unzulässig.

KC_SC_LISTENER_PORT_MISSING (nur auf Unix-, Linux- und Windows-Systemen)

Es wurde kein *listener_port* angegeben.

KC_SC_INVALID_BCAMAPPL_PORT (nur auf Unix-, Linux- und Windows-Systemen)

Die angegebene Portnummer ist ungültig.

Returncodes bei obj_type = KC_KSET:

Maincode = KC_MC_REJECTED Der Aufruf wurde von UTM abgewiesen. **Subcode:**
KC_SC_INVALID_KEY_VALUE Es wurde versucht mehr Keys zu erzeugen, als der in der Anwendung generierte Maximalwert erlaubt.

Returncodes bei obj_type = KC_LSES:

Maincode = KC_MC_REJECTED Der Aufruf wurde von UTM abgewiesen. **Subcode:**
KC_SC_LPAP_MISSING Es wurde kein LPAP-Partner angegeben.

Returncodes bei obj_type = KC_LTAC:

Maincode = KC_MC_REJECTED Der Aufruf wurde von UTM abgewiesen. **Subcodes:**
KC_SC_INVALID_WAITTIME Dem Parameter *waittime* wurde eine negative Wartezeit zugewiesen.
KC_SC_INVALID_LTACUNIT Dem Parameter *ltacunit* wurde ein Wert kleiner als `0` oder größer als `4095` zugewiesen.
KC_SC_INVALID_LOCK Der in der LTAC-Anweisung angegebene *lockcode* ist kleiner als `0` oder größer als der erlaubte Maximalwert (KDCDEF-Anweisung MAX, Operand KEYVALUE).
KC_SC_NOT_ALLOWED *lock_code* und *access_list* dürfen nicht zusammen angegeben werden.
KC_SC_INVALID_ACL Das angegebene Keyset existiert nicht.
KC_SC_INVALID_RTAC Bei *code*=INTEGER: Der Wert für *recipient_TPSU_title* ist größer als maximal zulässig. Bei *code*=PRINTABLE-STRING: Der RTAC-Name ist falsch.
KC_SC_LPAP_NOT_EXISTENT Der angegebene LPAP-, OSI-LPAP- oder Master-LPAP-Partner existiert nicht.
KC_SC_KSET_DEL Das über *access_list* referenzierte Keyset wurde gelöscht.
KC_SC_NAME_TOO_LONG Der dem Parameter *rtac* zugewiesene Name ist zu lang.
KC_SC_NAME_TOO_SHORT Der dem Parameter *rtac* zugewiesene Name ist zu kurz.
KC_SC_INVALID_CHAR_IN_STRING Der RTAC-Name ist falsch.

Returncodes bei obj_type = KC_LTERM:

Maincode = KC_MC_OK

Der Aufruf wurde fehlerfrei bearbeitet.

Subcode:

KC_SC_INVALID_LEVEL

Sie haben in *plev* und/oder *qlev* Werte angegeben, die die erlaubten Maximalwerte überschreiten. Der angegebene Wert wurde durch den Standardwert ersetzt.

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen.

Subcodes:

KC_SC_INVALID_NAME

Der angegebene Name des Objektes beginnt mit 'KDC'. Zu Format der Objektnamen siehe Abschnitt „Format und Eindeutigkeit der Objektnamen".

KC_SC_NAME_EXISTENT

Der für das zu erzeugende Objekt angegebene Name existiert bereits als TAC-Name.

KC_SC_INVALID_FORMAT

Das in *format_name* angegebene Format ist ein #Format, es ist jedoch kein Anmelde-Vorgang generiert (es existiert kein TAC mit dem Namen KDCSGNTC).

KC_SC_NO_FORMAT_ALLOWED

In *format_name* und *format_attr* wurde ein Startformat angegeben, es ist jedoch kein Formatierungssystem generiert (KDCDEF-Steueranweisung FORMSYS).

KC_SC_INVALID_FORMAT_USAGE

In *format_name*, *format_attr* ist ein Startformat angegeben, obwohl *usage_type* ='O' gesetzt ist.

KC_SC_INVALID_PLEV_RESTART

Es ist *plev* > '0' und *restart* ='N' gesetzt.

KC_SC_INVALID_PLEV_QAMSG

Es ist *plev* > '0' und *qamsg*='N' gesetzt.

KC_SC_INVALID_PLEV_USAGE

Es ist *plev* > '0' und *usage_type*='D' gesetzt.

KC_SC_INVALID_RESTART_QAMSG

Es ist *restart*='N' und *qamsg*='Y' gesetzt.

KC_SC_KSET_NOT_EXISTENT

Zu dem in *kset* angegebenen Namen existiert kein Keyset.

KC_SC_INVALID_USAGE_CTERM

Dem LTERM-Partner soll ein Druckersteuer-LTERM zugeordnet werden (Angabe in *cterm*), obwohl *usage_type*='D' gesetzt wurde (Dialog-Partner).

KC_SC_CTERM_NOT_EXISTENT

Der in *cterm* (Druckersteuer-LTERM) angegebene Name existiert nicht.

KC_SC_CTERM_DEL

Der LTERM-Partner, der zu dem in *cterm* angegebenen Namen gehört, wurde gelöscht.

KC_SC_INVALID_CTERM_USAGE

Der LTERM-Partner, der zu dem in *cterm* angegebenen Namen gehört, ist kein Dialog-Partner (*usage_type*='D').

KC_SC_INVALID_USER_USAGE

Dem LTERM-Partner soll eine Benutzerkennung zugeordnet werden (Angabe in *user_gen*); *usage_type* wird jedoch auf 'O' (Drucker) gesetzt.

KC_SC_USER_NOT_ALLOWED

Im Feld *user_gen* ist eine Benutzerkennung angegeben, die Anwendung ist aber ohne Benutzerkennungen generiert.

KC_SC_KSET_DEL

Das referenzierte Keyset wurde gelöscht.

KC_SC_USER_NOT_EXISTENT

Die in *user_gen* angegebene Benutzerkennung existiert nicht; die Anwendung ist mit Benutzerkennungen generiert.

KC_SC_USER_DEL

Die in *user_gen* angegebene Benutzerkennung wurde gelöscht.

KC_SC_USER_NOT_ADMINISTRABLE

Die in *user_gen* angegebene Benutzerkennung ist nicht administrierbar, z.B. weil es eine von UTM intern erzeugte Benutzerkennung ist.

KC_SC_USER_ALREADY_EXISTS

Die Anwendung wurde ohne Benutzerkennungen generiert.
Mit dem Namen, den Sie in *lt_name* angegeben haben (Name des LTERM-Partners), existiert bereits eine von UTM implizit erzeugte Benutzerkennung.

KC_SC_CTERM_IS_TPOOL

Das in *cterm* angegebene Objekt ist ein LTERM-Partner, der zu einem LTERM-Pool gehört. Er kann nicht als Druckersteuer-LTERM angegeben werden.

KC_SC_CTERM_IS_MUX (nur auf BS2000-Systemen)

Das in *cterm* angegebene Objekt ist ein LTERM-Partner, der zu einem Multiplexanschluss gehört. Er kann nicht als Druckersteuer-LTERM angegeben werden.

KC_SC_CTERM_IS_UTM_D

Der in *cterm* angegebene Name gehört zu einem LPAP- oder OSI-LPAP-Partner für den Anschluss von Partner-Servern.

KC_SC_INVALID_LOCK

Der in *lock_code* angegebene Lockcode liegt nicht zwischen 1 und dem in der Anwendung erlaubten Maximalwert (KDCDEF-Anweisung MAX, Operand KEYVALUE).

KC_SC_INVALID_BUNDLE_CTERM

Das angegebene CTERM ist ein Master oder Slave eines LTERM-Bündels.

KC_SC_PRINCIPAL_AND_KERBEROS

Der Wert 'Y' in *kerberos _dialog* ist nicht erlaubt, wenn sowohl MAX PRINCIPAL-LTH als auch MAX CARDLTH den Wert 0 haben.

Returncodes bei obj_type = KC_PROGRAM:

Maincode = KC_MC_REJECTED Der Aufruf wurde von UTM abgewiesen. **Subcodes:**
KC_SC_LMOD_MISSING In *load_module* wurde kein Lademodul/Shared Object/DLL angegeben.
KC_SC_COMP_MISSING (nur auf BS2000-Systemen) In *compiler* wurde kein Compiler angegeben.
KC_SC_LMOD_NOT_EXISTENT Das in *load_module* angegebene Lademodul/Shared Object/DLL existiert nicht.
KC_SC_LMOD_NOT_CHANGEABLE Das in *load_module* angegebene Lademodul/Shared Object/DLL ist nicht austauschbar.
KC_SC_NO_LMOD Die Anwendung wurde nicht mit Lademodulen/Shared Object/DLLs generiert. Es kann kein Teilprogramm mit KC_CREATE_OBJECT dynamisch in die Konfiguration aufgenommen werden.
KC_SC_COMP_NOT_GEN Die Anwendung enthält kein Sprachanschlussmodul, das dem in *compiler* angegebenen Compiler entspricht.
KC_SC_KDCADM_ONCALL_LMOD Das Standard-Administrationsprogramm KDCADM darf nicht mit Lademodus ONCALL erzeugt werden.
KC_SC_MFCOBOL_AND_NETCOBOL (nur auf Unix- und Linux-Systemen) In einer UTM-Anwendung dürfen nicht gleichzeitig Programme für MFCOBOL (Micro Focus COBOL) und NETCOBOL eingesetzt werden.
KC_SC_LANG_ENV_MISSING (nur auf Unix- und Linux-Systemen) Es ist keine Sprachumgebung für MFCOBOL oder NETCOBOL vorhanden.

Returncodes bei obj_type = KC_PTERM:

Maincode = KC_MC_OK

Der Aufruf wurde fehlerfrei bearbeitet.

Subcodes:

KC_SC_INVALID_USAGE_APPLI_UPIC

Die Angaben in *ptype* und *usage_type* passen nicht zusammen. Es wurde *ptype*='UPIC-...' zusammen mit *usage_type*='O' angegeben. Der Wert in *usage_type* wurde automatisch durch 'D' ersetzt.

KC_SC_INVALID_IDLETIME

Der Wert des Parameters *idletime* wurde verändert, weil Sie einen Wert zwischen 1 und 59 angegeben haben. UTM hat *idletime* auf den kleinsten zulässigen Wert gesetzt.

KC_SC_INVALID_PROTOCOL

Die Angaben in *ptype* und *protocol* passen nicht zusammen. Folgende Fälle sind zu unterscheiden:

- Es wurde *ptype*='UPIC-...' oder '*RSO' und *protocol*='S' angegeben. Der Wert in *protocol* wurde automatisch durch 'N' ersetzt.
- Es wurde *ptype*='*ANY' und *protocol* ='N' angegeben. Der Wert in *protocol* wurde automatisch durch 'S' ersetzt.

KC_SC_INVALID_USAGE_AND_PROT

Die Angaben in *ptype*, *protocol* und *usage_type* passen nicht zusammen. Es wurde *ptype*='UPIC-...' zusammen mit *usage_type*='O' und *protocol*='S' angegeben. Der Wert in *usage_type* wurde automatisch durch 'D', der Wert in *protocol* durch 'N' ersetzt.

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen.

Subcodes:

KC_SC_PROCESSOR_MISSING (nur auf BS2000-Systemen)

In *pronam_long* wurde kein Rechnername angegeben. Die Angabe von *pronam_long* ist in UTM-Anwendungen auf BS2000-Systemen Pflicht.

KC_SC_PTYPE_MISSING (nur auf BS2000-Systemen)

In *ptype* wurde kein Partnertyp angegeben. Die Angabe ist in UTM-Anwendungen auf BS2000-Systemen Pflicht.

KC_SC_PROCESSOR_NOT_ALLOWED

In *pronam_long* wurde ein Rechnername länger als 8 Zeichen angegeben, der keinen Punkt ('.') enthält und somit kein DNS-Name sein kann, oder auf Unix-, Linux- und Windows-Systemen wurde in *pronam_long* ein Rechnername angegeben, obwohl *ptype*='TTY', 'PRINTER' oder 'UPIC-L' gesetzt wurde.

KC_SC_INVALID_NAME

Der angegebene Objektname beginnt mit „KDC“. Dieser Name ist für UTM reserviert. Zum Format der Objektnamen siehe Abschnitt „Format und Eindeutigkeit der Objektnamen".

KC_SC_INVALID_STATUS_CONNECT

Es wurde *state*='N' zusammen mit *auto_connect*='Y' angegeben.

KC_SC_INVALID_PROTOCOL_USAGE

Es wurde *protocol*='N' und *usage*='O' angegeben und *ptype* wurde weder 'RSO' noch 'APPLI' noch 'SOCKET' zugewiesen.

KC_SC_INVALID_CID_USAGE

In *cid* wurde eine Drucker-Id angegeben, obwohl *usage_type*='D' (auf BS2000-Systemen) bzw. *ptype*='tty' (Unix-, Linux- und Windows-Systeme) gesetzt wurde.

KC_SC_BCAMAPPL_NOT_EXISTENT

Der in *bcamappl* angegebene Anwendungsname existiert nicht.

KC_SC_INVALID_BCAMAPPL_PORT (nur auf Unix-, Linux- und Windows-Systemen)

Ungültiger Listener-Port.

KC_SC_INVALID_BCAMAPPL_PTYPE

Der in *bcamappl* angegebene Name ist nicht identisch mit dem in der KDCDEF-Steueranweisung MAX definierten Anwendungsnamen (APPLINAME), obwohl *ptype* != 'APPLI', 'SOCKET' oder 'UPIC-R' ist.

KC_SC_LTERM_NOT_EXISTENT

Der in *lterm* angegebene LTERM-Partner existiert nicht.

KC_SC_PTYPE_NO_LTERM

Es wurde *ptype*='APPLI', 'SOCKET' oder 'UPIC-...' gesetzt, aber in *lterm* wurde kein LTERM-Partner angegeben.

KC_SC_INVALID_USAGE_LTERM

Die Angabe in *usage_type* passt nicht zu dem in *lterm* angegebenen LTERM-Partner.

KC_SC_INVALID_BUNDLE_USAGE

usage_type='O' nicht erlaubt bei Bündel.

KC_SC_INVALID_BUNDLE

Es wurde *usage_type*='D' gesetzt und in *lterm* wurde ein LTERM-Partner angegeben, dem bereits ein Client zugeordnet ist.

KC_SC_INVALID_GROUP_USAGE

usage_type='O' nicht erlaubt bei Gruppe.

KC_SC_INVALID_PROV_BUNDLE

Es wurde *usage_type*='D' gesetzt und in *lterm* wurde ein LTERM-Partner angegeben, dem in der gleichen Transaktion bereits ein Client zugeordnet wurde.

KC_SC_LTERM_DEL

Der in *lterm* angegebene LTERM-Partner wurde gelöscht.

KC_SC_CID_MISSING

Angabe in *cid* fehlt:
Dem in *lterm* angegebenen LTERM-Partner ist ein Druckersteuer-LTERM (Angabe in *cterm*) zugeordnet. Für den Drucker muss dann eine Drucker-Id angegeben werden.

KC_SC_INVALID_CID

Die in *cid* angegebene Drucker-Id gehört bereits zu einem anderen Drucker, der demselben Druckersteuer-LTERM zugeordnet ist.

KC_SC_CTERM_DEL

Das Druckersteuer-LTERM des in *lterm* angegebenen LTERM-Partners wurde gelöscht.

KC_SC_USRT_TAB_FULL

Bei *ptype*='APPLI', 'SOCKET' oder 'UPIC-...': UTM kann keine Verbindungsbenutzerkennung erzeugen, da die bei der Generierung reservierten Tabellenplätze für Benutzerkennungen ausgeschöpft sind.

KC_SC_PROCESSOR_NOT_ALLOWED (nur auf Unix-, Linux- und Windows-Systemen)

In *pronam* wurde ein Rechnername angegeben, obwohl *ptype*='TTY', 'PRINTER' oder 'UPIC-L' gesetzt wurde.

KC_SC_INVALID_MAP_PTYPE (nur auf Unix-, Linux- und Windows-Systemen)

Es wurde *map* != 'U' gesetzt, obwohl *ptype* != 'APPLI' oder 'SOCKET' angegeben wurde.

KC_SC_INVALID_MAP_AND_PROT (nur auf BS2000-Systemen)

Es wurde *map* != 'U' gesetzt, obwohl *ptype* != 'SOCKET' angegeben wurde.

KC_SC_INVALID_CONNECT_PTYPE (nur auf Unix-, Linux- und Windows-Systemen)

Es wurde *auto_connect*='Y' zusammen mit *ptype*='TTY' oder 'UPIC-...' angegeben.

KC_SC_INVALID_AUTOUSER_PTYPE

ptype='APPLI', 'SOCKET' oder 'UPIC-...':
Die zu dem in *lterm* angegebenen LTERM-Partner definierte Verbindungsbenutzerkennung (*user_gen*) wird nicht von der eigenen Transaktion erzeugt.

KC_SC_INVALID_LTERM_PTYPE

ptype='APPLI', 'SOCKET' oder 'UPIC-...':
Der in *lterm* angegebene LTERM-Partner wird nicht in derselben Transaktion erzeugt.

KC_SC_LTERM_IS_TPOOL

Der in *lterm* angegebene LTERM-Partner gehört zu einem LTERM-Pool.

KC_SC_LTERM_IS_MUX (nur auf BS2000-Systemen)

Der in *lterm* angegebene LTERM-Partner gehört zu einem Multiplexanschluss, d.h. er wurde von UTM implizit für einen Multiplexanschluss erzeugt.

KC_SC_LTERM_IS_UTM_D

Der in *lterm* angegebene Name gehört zu einem LPAP- oder OSI-LPAP-Partner für den Anschluss von Partner-Servern.

KC_SC_LTERM_IS_MASTER

Das angegebene LTERM ist ein Master-Lterm.

KC_SC_LTERM_IS_ALIAS

Das angegebene LTERM ist ein Alias-Lterm.

KC_SC_INVALID_GROUP_PTYPE

Das angegebene LTERM ist ein Primary Lterm und der PTYPE ist nicht APPLI oder SOCKET.

KC_SC_INVALID_LTERM_SLAVE_PTYP

Das angegebene LTERM ist ein Slave-Lterm und der PTYPE ist nicht APPLI oder SOCKET.
Auf BS2000-Systemen: Unterschiedliche PTYPEs innerhalb eines Bündels.

KC_SC_INVALID_APPLI_USER

Bei *ptype*='APPLI', 'SOCKET' oder 'UPIC-R': Für den im Feld *lterm* angegebenen LTERM-Partner wurde keine Verbindungsbenutzerkennung angegeben, d.h. beim Eintragen des LTERM-Partners wurde *user_gen* nicht angegeben. Eine Benutzerkennung mit dem Namen des LTERM-Partners existiert, wurde aber nicht in derselben Transaktion erzeugt wie der Client (siehe Abschnitt "Clients, Drucker und LTERM-Partner eintragen").

KC_SC_INVALID_LISTENID (nur auf BS2000-Systemen)

Die in *listener_port* angegebene Nummer ist unzulässig.

KC_SC_PRONAM_NOT_RSO (nur auf BS2000-Systemen)

In ptype wurde 'RSO' angegeben, pronam_long wurde aber nicht mit '*RSO' belegt.

KC_SC_PTYPE_NOT_RSO (nur auf BS2000-Systemen)

In pronam_long wurde 'RSO' angegeben, ptype wurde aber nicht mit '*RSO' belegt.

KC_SC_INVALID_USAGE _APPLI_UPIC

Es wurde ptype='APPLI', 'SOCKET' oder 'UPIC-...' zusammen mit USAGE='O' angegeben.

KC_SC_INVALID_IDLETIME_USAGE

idletime wurde für eine Ausgabestation angegeben.

KC_SC_INVALID_AUTOUSER_PTYPE

Es wurde *ptype* ='APPLI', 'SOCKET' oder 'UPIC-...' angegeben, aber der USER mit dem Namen des angegebenen LTERM wird nicht von der eigenen Transaktion erzeugt.

KC_SC_PRINTER_NT_NOT_SUPPORTED (nur auf Windows-Systemen)

In einer UTM-Anwendung auf Windows-Systemen wurde *ptype*='PRINTER' gesetzt.
openUTM auf Windows-Systemen unterstützt jedoch keine Drucker.

KC_SC_INVALID_PTYPE_AND_PROT

Das PTERM ist nicht mit *ptype*='SOCKET' und das referenzierte BCAMAPPL mit TCP/IP generiert.BS2000-Systeme:
Das PTERM ist mit *ptype*='SOCKET' und das referenzierte BCAMAPPL nicht mit TCP/IP generiert.

KC_SC_INVALID_TPROT_AND_TPROT (nur auf Unix, Linux- und Windows-Systemen)

Das PTERM ist mit *ptype*='SOCKET' und das referenzierte BCAMAPPL nicht mit TCP/IP generiert.

KC_SC_INVALID_USP_AND_PROT

Im Feld *usp_hdr* steht ein Wert ungleich NO und das referenzierte BCAMAPPL verfügt nicht über TCP/IP.

KC_SC_TPROT_NOT_ALLOWED

Transportprotokoll nicht erlaubt.

KC_SC_KEY_NOT_GEN_CREA_IT

Im Feld encryption_level wurde eine Verschlüsselungsebene gewählt, für die bei der Generierung kein RSA-Schlüsselpaar eingerichtet wurde. Wenn ein PTERM mit dieser Verschlüsselungsebene erzeugt werden soll, dann müssen Sie erst dynamisch ein RSA-Schlüsselpaar mit dem gewünschten Level erzeugen. Bitte beachten Sie, dass dies bei einer Verschlüsselungsebene höher als 2 längere Zeit in Anspruch nehmen kann.

Returncodes bei obj_type = KC_TAC:

Maincode = KC_MC_OK

Der Aufruf wurde fehlerfrei bearbeitet.

Subcode:

KC_SC_INVALID_VALUE

Einer oder mehrere der folgenden Werte waren ungültig oder wurden automatisch gesetzt:

- In *qlev* wurde eine Anzahl angegeben, die größer als der erlaubte Maximalwert ist. UTM hat den Wert durch den Maximalwert ersetzt.
- In *cpu_time_msec* wurde eine Zeit zwischen '1' und '999' msec angegeben. Die Zeit wurde durch '1000' ersetzt.
- In *cpu_time_msec* wurde eine Zeit angegeben, die größer als der erlaubte Maximalwert ist. Der Wert wurde durch den Maximalwert ersetzt.
- In *real_time_sec* wurde eine Zeit angegeben, die größer als der erlaubte Maximalwert ist. Der Wert wurde durch den Maximalwert ersetzt.
- In *runprio* wurde eine Priorität zwischen '1' und '29' angegeben. Der Wert wurde durch '30' ersetzt.
- In *tacunit* wurde ein Wert angegeben, der größer als der erlaubte Maximalwert ist. Der Wert wurde durch den Maximalwert ersetzt.

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen.

Subcodes:

KC_SC_NOT_ALLOWED

Die Angabe von *lock_code* und *access_list* zusammen ist nicht erlaubt.

KC_SC_PROGRAM_MISSING

Es wurde keine Angabe für *program* gemacht.

KC_SC_INVALID_TYPE

Bei UTM-FF sind keine Queues erlaubt.

KC_SC_INVALID_NAME

Sie wollten einen Administrations-TAC erzeugen, ohne admin='Y' zu setzen, oder der angegebene TAC-Name (*tc_name*) beginnt mit 'KDC' (diese Namen sind für UTM reserviert).
Zum Format der Objektnamen siehe Abschnitt „Format und Eindeutigkeit der Objektnamen".

KC_SC_TACUNIT_ILL

Ungültiger Wert für *tacunit*.

KC_SC_PROGRAM_NOT_EXISTENT

Das in *program* angegebene Teilprogramm existiert nicht.

KC_SC_INVALID_EXIT_PROGRAM

Der in *exit_name* angegebene VORGANG-Exit gehört zu einem Lademodul, Shared Object, DLL, das mit Lademodus ONCALL generiert ist. Das in *program* angegebene Teilprogramm ist jedoch nicht in diesem Lademodul enthalten.

KC_SC_NAME_EXISTENT

Der in *tc_name* angegebene Transaktionscode ist bereits als Name eines LTERM-Partners definiert. Die Namen von Transaktionscodes und LTERM-Partnern gehören zu derselben Namensklasse (siehe Abschnitt „Format und Eindeutigkeit der Objektnamen").

KC_SC_EXIT_NEXT_TAC

In *exit_name* wurde ein VORGANG-Exit angegeben, obwohl der Transaktionscode als Folge-TAC konfiguriert werden soll (*call_type*='N').

KC_SC_PROGRAM_DEL

Das in *program* angegebene Teilprogramm wurde gelöscht.

KC_SC_EXIT_NOT_EXISTENT

Der in *exit_name* angegebene VORGANG-Exit existiert nicht.

KC_SC_INVALID_TCBENTRY

Die Angabe von *tcbentry* ist nicht erlaubt.

KC_SC_EXIT_DELETED

Der in *exit_name* angegebene VORGANG-Exit ist gelöscht.

KC_SC_XOPEN_NOT_ALLOWED

In *api* wurde ein Wert ungleich 'K' (KDCS) angegeben und die Anwendung wurde ohne X/Open-TACs generiert. Sie können einen Transaktionscode für ein Teilprogramm, das die Funktionen der X/Open-Programmschnittstellen nutzt, nur dynamisch konfigurieren, wenn mindestens ein Transaktionscode dieses Typs mit KDCDEF statisch generiert wurde.

KC_SC_INVALID_QMODE

q_mode='W' ist nur für TAC-Queues erlaubt.

KC_SC_INVALID_QMODE_QLEV

q_mode='W' aber *qlev* ist nicht zwischen 1 und 32766.

KC_SC_INVALID_QMODE_FF

Ungültiger *q_mode* für UTM-FF.

KC_SC_KSET_DEL

Das über *kset* oder über *access_list* referenzierte Keyset wurde gelöscht.

KC_SC_READ_ACL_DEL

Das über *q_read_acl* referenzierte Keyset wurde gelöscht.

KC_SC_WRITE_ACL_DEL

Das über *q_write_acl* referenzierte Keyset wurde gelöscht.

KC_SC_INVALID_LOCK

Der in *lock_code* angegebene Lockcode liegt nicht zwischen 1 und dem in der Anwendung erlaubten Maximalwert (KEYVALUE-Operand der MAX-Anweisung).

KC_SC_INVALID_TACCLASS

Die Angaben in *tacclass* und *tac_type* passen nicht zusammen:

- Es wurde *tac_type*='D' (Dialog-TAC) gesetzt und in *tacclass* wurde ein Wert angegeben, der nicht zwischen '1' und '8' liegt.
- Es wurde *tac_type*='A' (Asynchron-TAC) gesetzt und in *tacclass* wurde ein Wert angegeben, der nicht zwischen '9' und '16' liegt.

KC_SC_NO_TACCLASS_GENERATED

Im Feld *tacclass* wurde eine Angabe gemacht, die Anwendung wurde aber ohne TAC-Klassen generiert.

KC_SC_PGWT_TACCLASS

Im Feld *pgwt* wurde 'Y' angegeben; das ist nur erlaubt, wenn bei der KDCDEF-Generierung die Anweisung TAC-PRIORITIES abgesetzt wurde.

KC_SC_PGWT_NO_PGWT_TASKS

Im Feld *pgwt* wurde 'Y' angegeben, bei der KDCDEF-Generierung der Anwendung wurde jedoch MAX TASKS-IN-PGWT=0 (Standardwert) angegeben.

KC_SC_ILLEGAL_STATUS

Im Feld *state* wurde 'K' (Keep) angegeben, obwohl *tac_type*='D' (d.h. der Transaktionscode ist kein Asynchron-TAC) und/oder *call_type* != 'F'oder 'B' (der Transaktionscode ist nicht als 1. TAC eines Vorgangs definiert).

KC_SC_PGWT_YES_NO_TACCLASS

Sie haben im Feld *pgwt* 'Y' angegeben, obwohl die Anwendung ohne TAC-Klassen generiert ist.

KC_SC_CALLTYPE_N_ENCRYPT

Sie haben im Feld *encryption_level* ungleich 'N' angegeben, obwohl der TAC kein Vorgangs-TAC ist, d.h. *call_type*='N'.

KC_SC_INVALID_READ_ACL

Das in *q_read_acl* angegebene Keyset existiert nicht.

KC_SC_INVALID_WRITE_ACL Das in *q_write_acl* angegebene Keyset existiert nicht.
KC_SC_INVALID_ACL Das in *access_list* angegebene Keyset existiert nicht.
KC_SC_DLETQ_YES_NOT_ALLOWED Ungültiger Wert für *dead_letter_q*.

Returncodes bei obj_type = KC_USER:

Maincode = KC_MC_OK

Der Aufruf wurde fehlerfrei bearbeitet.

Subcode:

KC_SC_INVALID_PROTECT_PW

Die Angabe in *protect_pw16_lth* und/oder die Angabe in *protect_pw_time* war größer als der erlaubte Maximalwert. Die Angabe wurde durch den Maximalwert ersetzt.

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen.

Subcodes:

KC_SC_CARD_TAB_FULL

Der bei der KDCDEF-Generierung reservierte Tabellenplatz für CARD ist bereits belegt oder es wurden keine Tabellenplätze für CARD reserviert.

KC_SC_NO_CARD_ALLOWED (nur auf Unix, Linux- und Windows-Systemen)

Es wurde *card...* angegeben, es ist jedoch kein Formatierungssystem generiert.

KC_SC_INVALID_NAME

Die angegebene Benutzerkennung (*us_name*) beginnt mit 'KDC'. Diese Namen sind für UTM reserviert. Zum Format der Objektnamen siehe Abschnitt „Format und Eindeutigkeit der Objektnamen".

KC_SC_INVALID_FORMAT (nur auf BS2000-Systemen)

Das in *format_name* und *format_attr* angegebene Startformat ist ein #Format, es ist jedoch kein Anmelde-Vorgang generiert (es existiert kein TAC mit dem Namen KDCSGNTC).

KC_SC_NO_FORMAT_ALLOWED (nur auf BS2000-Systemen)

In *format_name* und *format_attr* wurde ein Startformat angegeben, es ist jedoch kein Formatierungssystem generiert (KDCDEF-Steueranweisung FORMSYS).

KC_SC_COMPL_MISSING

Das in *password* angegebene Passwort erfüllt nicht die in *protect_pw_compl* geforderte Komplexitätsstufe.

KC_SC_KSET_NOT_EXISTENT

Zu dem in *kset* angegebenen Namen existiert kein Keyset.

KC_SC_INVALID_POSITION (nur auf BS2000-Systemen)

Die Angabe im Feld *card_position* ist ungültig.

KC_SC_MIN_LTH_WITHOUT_PASSWORD

In *password16* wurde kein Passwort angegeben, obwohl *protect_pw16_lth* > '0' gesetzt ist.

KC_SC_APPLICATION_WITHOUT_USER

Sie können keine Benutzerkennung erzeugen, da die Anwendung ohne Benutzerkennungen generiert wurde.

KC_SC_INVALID_READ_ACL

Das in *q_read_acl* angegebene Keyset existiert nicht.

KC_SC_INVALID_WRITE_ACL

Das in *q_write_acl* angegebene Keyset existiert nicht.

KC_SC_INVALID_QMODE_QLEV

q_mode='W', aber *qlev* ist nicht zwischen 1 und 32766.

KC_SC_INVALID_QMODE_FF

Ungültiger *q_mode* für UTM-FF.

KC_SC_KSET_DEL

Das über *kset* referenzierte Keyset wurde gelöscht.

KC_SC_READ_ACL_DEL

Das über *q_read_acl* referenzierte Keyset wurde gelöscht.

KC_SC_WRITE_ACL_DEL

Das über *q_write_acl* referenzierte Keyset wurde gelöscht.

KC_SC_INVALID_PRINCIPAL (nur auf BS2000-Systemen)

Es wurde ein Principal angegeben und gleichzeitig der Parameter CARD oder Password spezifiziert.

KC_SC_INVALID_QLEV_FF

Ungültiger *qlev* für UTM-FF.

KC_SC_PRINCIPAL_AND_PW (nur auf BS2000-Systemen)

Es ist nicht möglich, einen USER sowohl mit Principal als auch mit Passwort zu erzeugen.

KC_SC_PRINCIPAL_AND_CARD (nur auf BS2000-Systemen)

Es ist nicht möglich, einen USER sowohl mit Principal als auch mit Chip-Karte zu erzeugen.

KC_SC_PRINCIPAL_TABLE_FULL (nur auf BS2000-Systemen)

Der bei der KDCDEF-Generierung reservierte Tabellenplatz für PRINCIPAL ist bereits belegt oder es wurden keine Tabellenplätze für PRINCIPAL reserviert.

KC_SC_PRINCIPAL_TOO_LONG (nur auf BS2000-Systemen)

Der Principal ist länger als der in MAX PRINCIPAL-LTH angegebene Wert.
KC_SC_INVALID_CLUSTER_RESTART Nur für UTM-Cluster-Anwendungen: Ungültiger Wert für *restart*.

11.2.4 KC_CREATE_STATEMENTS - KDCDEF-Steueranweisungen erzeugen (inverser KDCDEF)

Mit KC_CREATE_STATEMENTS können Sie während des Anwendungslaufs (online) einen inversen KDCDEF-Lauf starten. Der inverse KDCDEF erzeugt KDCDEF-Steueranweisungen aus den Konfigurationsdaten. Damit können alle Änderungen, die durch das dynamische Eintragen, Modifizieren und Löschen von Objekten bewirkt wurden, für eine Neugenerierung übernommen werden.

Die vom inversen KDCDEF erzeugten KDCDEF-Steueranweisungen stellen im folgenden Sinne einen konsistenten Stand der Konfiguration der laufenden Anwendung dar:

Die von einer Transaktion durchgeführten Änderungen der Konfigurationsdaten werden von einem gleichzeitig laufenden inversen KDCDEF immer vollständig berücksichtigt.

Siehe auch openUTM-Handbuch „Anwendungen generieren“, Abschnitt „Ablauf eines inversen KDCDEF-Laufs“.

Mit dem inversen KDCDEF können Sie folgende KDCDEF-Steueranweisungen erzeugen:

- CON-Anweisungen für Transportverbindungen zu entfernten LU6.1-Anwendungen.
- KSET-Anweisungen für alle Keysets.
- LSES-Anweisungen für alle LU6.1-Sessions.
- LTAC-Anweisungen für Transaktionscodes, über die Service-Programme in Partner-Anwendungen gestartet werden.
- LTERM-Anweisungen für alle LTERM-Partner, die nicht zu einem LTERM-Pool bzw. Multiplexanschluss gehören.
- PTERM-Anweisungen für alle Clients und Drucker, die explizit in die Konfiguration eingetragen wurden.
- PROGRAM-Anweisungen für alle Teilprogramme und VORGANG-Exits.
- TAC-Anweisungen für alle Transaktionscodes und TAC-Queues der Anwendung.
- USER-Anweisungen für alle Benutzerkennungen einschließlich ihrer Queues.

Der inverse KDCDEF erzeugt eine Steueranweisung für jedes Objekt des angegebenen Objekttyps, das in der Konfiguration der Anwendung enthalten ist, unabhängig davon, ob es dynamisch eingetragen wurde oder nicht und ob seine Eigenschaften modifiziert wurden oder nicht. Für Objekte, die mit KC_DELETE_OBJECT gelöscht wurden, erzeugt der inverse KDCDEF keine Steueranweisungen.

Detaillierte Angaben zum inversen KDCDEF finden Sie im Kapitel „Generierungsanweisungen aus der KDCFILE erzeugen".

Steuern des inversen KDCDEF-Laufs

Der inverse KDCDEF unterscheidet die folgenden sieben Objektgruppen:

1.Gruppe	LTERM-Partner, Clients, Drucker (Objekttypen: KC_LTERM, KC_PTERM)
2.Gruppe	Teilprogramme, Transaktionscodes, TAC-Queues (Objekttypen: KC_PROGRAM, KC_TAC)
3.Gruppe	Benutzerkennungen (Objekttyp: KC_USER)
4.Gruppe	Keysets (Objekttyp: KC_KSET)
5.Gruppe	Transaktionscodes, über die Service-Programme in Partner-Anwendungen gestartet werden (Objekttyp: KC_LTAC)
6.Gruppe	Transportverbindungen zu LU6.1-Anwendungen (Objekttyp: KC_CON)
7.Gruppe	LU6.1-Sessions (Objekttyp: KC_LSES)

Mit dem KC_CREATE_STATEMENTS-Aufruf können Sie die KDCDEF-Steueranweisungen für Objekte einer oder mehrerer dieser Gruppen erzeugen.

Beim KC_CREATE_STATEMENTS-Aufruf müssen Sie angeben, in welche Datei UTM die KDCDEF-Steueranweisungen schreiben soll. Sie können alle Steueranweisungen in eine Datei schreiben lassen oder für jede Objektgruppe eine eigene Datei angeben. Beim Aufruf können Sie darüber hinaus angeben, ob UTM neue Dateien anlegen oder bereits existierende Dateien erweitern soll.
Auf BS2000-Systemen können die Steueranweisungen statt in eine Datei auch in ein LMS-Bibliothekselement geschrieben werden. Das Vorgehen bei Bibliothekselementen ist analog zum Vorgehen bei Dateien.

Ablauf eines inversen KDCDEF-Laufs

Der Zeitpunkt, zu dem der inverse KDCDEF-Lauf gestartet wird, und der Ablauf, sind abhängig vom momentanen Zustand der Anwendung. Die folgenden zwei Fälle sind zu unterscheiden:

- Der inverse KDCDEF-Lauf wird asynchron gestartet, wenn zum Zeitpunkt des KC_CREATE_STATEMENTS-Aufrufs Transaktionen laufen, die schreibend auf die Konfigurationsdaten der Objekte zugreifen. Der inverse KDCDEF-Lauf wird dann erst gestartet, wenn diese Transaktionen abgeschlossen sind. Bei neuen Transaktionen, die Daten der Objekttabellen verändern sollen, werden die entsprechenden Aufrufe zum Ändern der Konfigurationsdaten der Anwendung jedoch abgewiesen, bis der inverse KDCDEF-Lauf abgeschlossen, d.h. der Asynchron-Auftrag bearbeitet ist.

  In UTM-Cluster-Anwendungen (Unix-, Linux- und Windows-Systeme) gilt außerdem:
  Eine Cluster-global wirkende Administrationsaktion führt in jeder laufenden Knoten-Anwendung zu einer solchen Transaktion, die den Start des inversen KDCDEF verzögern kann. Umgekehrt kann die Ausführung einer globalen Administrationsaktion auf einem laufenden Knoten verzögert werden, wenn dort gerade ein inverser KDCDEF läuft.

- Der inverse KDCDEF-Lauf wird synchron gestartet, wenn zum Zeitpunkt des KC_CREATE_STATEMENTS-Aufrufs **keine** Transaktionen laufen, die schreibend auf die Konfigurationsdaten der Objekte zugreifen. Er ist bereits bei der Rückkehr in das Administrationsprogramm beendet. D.h. zu diesem Zeitpunkt sind bereits alle angeforderten KDCDEF-Steueranweisungen erzeugt und in den angegebenen Dateien abgelegt.

Ergebnis des inversen KDCDEF-Laufs

Nach einem erfolgreichen inversen KDCDEF-Lauf stehen die angeforderten Steueranweisungen in den beim Aufruf angegebenen Dateien. Diese Dateien können Sie bei einer Neugenerierung der Anwendung als Input für das UTM-Generierungstool KDCDEF verwenden. Jede der Dateien müssen Sie mit der KDCDEF-Steueranweisung OPTION DATA=Dateiname an KDCDEF übergeben. Die Dateien sind editierbar und können modifiziert werden.
Analoges gilt, wenn die Steueranweisungen auf BS2000-Systemen in LMS-Bibliothekselemente statt in Dateien geschrieben werden. Ob die Elemente editierbar sind, hängt allerdings von deren Typ ab; nur textartige Elemente sind modifizierbar.

Transaktionssicherung / Cluster

Der KC_CREATE_STATEMENTS-Aufruf greift nur lesend auf die Daten der KDCFILE zu. Er unterliegt deshalb nicht der Transaktionssicherung. Er kann nicht durch einen RSET-Aufruf in der gleichen Transaktion zurückgesetzt werden.

Für UTM-Cluster-Anwendungen (Unix-, Linux- und Windows-Systeme) gilt:
Der Aufruf wirkt Knoten-lokal, d.h. das Starten eines inversen KDCDEF-Laufs zum Erzeugen von Steueranweisungen aus den Konfigurationsdaten wird nur in dieser Knoten-Anwendung ausgeführt. Diese Knoten-lokale Wirkung reicht aus, da in jeder Knoten-Anwendung dieselben Objekte existieren. Eine Cluster-globale Wirkung würde lauter identische KDCDEF-Anweisungen erzeugen.

Wenn Knoten-Anwendungen mit unterschiedlichen Generierungen laufen (während eines Online-Updates), wird der Aufruf abgelehnt, da das Ergebnis ansonsten davon abhängig wäre, auf welcher Knoten-Anwendung der Aufruf ausgeführt wird.

Versorgung der zu übergebenden Bereiche

Funktion des Aufrufs	Angabe im			
	Parameterbereich	**Identifikationsbereich**	**Selektionsbereich**	**Datenbereich**
Online KDCDEF-Steueranweisungen erzeugen	Operationscode: KC_CREATE_ STATEMENTS	——	——	Datenstruktur mit Informationen zum Typ der zu erzeugenden Steueranweisungen sowie Namen und Schreibmodus der Dateien

Versorgung der Parameter	
Parameterbereich	
Feldname	Inhalt
version	KC_ADMI_VERSION_1
retcode	KC_RC_NIL
version_data	KC_VERSION_DATA_11
opcode	KC_CREATE_STATEMENTS
id_lth	0
select_lth	0
data_lth	Länge der Daten im Datenbereich
Identifikationsbereich	
—	
Selektionsbereich	
—	
Datenbereich	
Datenstruktur kc_create_statements_str	

KDCADMI-Aufruf
KDCADMI (&parameter_area, NULL, NULL, &data_area)

Rückgaben von UTM	
Parameterbereich	
Feldname	Inhalt
retcode	Returncodes

data_lth

In *data_lth* geben Sie die Länge der Datenstruktur *kc_create_statements_str* an.

Datenbereich

Im Datenbereich müssen Sie für jede der Objektgruppen angeben, ob UTM die zugehörigen KDCDEF-Steueranweisungen erzeugen soll oder nicht. Soll UTM Steueranweisungen für eine Objektgruppe erzeugen, dann müssen Sie zusätzlich die Datei angeben, in die UTM die Steueranweisungen schreiben soll, und den Schreibmodus der Datei. Zur Übergabe dieser Informationen an UTM ist in der Include-Datei *kcadminc.h* folgende Datenstruktur definiert.

Definition von Konstanten

```
#define KC_FILE_NAME_LTH    54
#define KC_ELEM_NAME_LTH    64
#define KC_VERSION_LTH      24
#define KC_TYPE_LTH          8
```

Definition der Indexkonstante

```
typedef enum
{  KC_DEVICE_STMT      = 0,
   KC_PROGRAM_STMT     = 1,
   KC_USER_STMT        = 2,
   KC_KSET_STMT        = 3,
   KC_LTAC_STMT        = 4,
   KC_CON_STMT         = 5,
   KC_LSES_STMT        = 6,
   KC_MAX_STMT_TYPE    = 6,
   KC_DUMMY_STMT_TYPE  = 7
} KC_INVDEF_TYPE;
```

Definition der Datenstruktur

```
struct kc_create_statements_str
{  struct
   {  char  create_control_stmts;
       char  file_name[KC_FILE_NAME_LTH];
       char  file_mode;
       char  lib_name[KC_FILE_NAME_LTH];
       char  elem_name[KC_ELEM_NAME_LTH];
       char  vers[KC_VERSION_LTH];
       char  type[KC_TYPE_LTH];

   } type_list[(int)KC_MAX_STMT_TYPE + 1];
  char stmt_type;
  char file_error_code[4];
};
```

Der Index KC_INVDEF_TYPE des Vektors *type_list* gibt die Gruppe der Objekte an:

KC_DEVICE_STMT

Steht für die 1. Gruppe, in der LTERM-Partner, Clients und Drucker zusammengefasst sind. In dieser Gruppe werden die KDCDEF-Steueranweisungen LTERM und PTERM erzeugt.

KC_PROGRAM_STMT

Steht für die 2. Gruppe, in der Teilprogramme, Transaktionscodes und TAC- Queues zusammengefasst sind. In dieser Gruppe werden die KDCDEF-Steueranweisungen PROGRAM und TAC erzeugt.

KC_USER_STMT

Steht für die 3. Gruppe der UTM-Benutzerkennungen. In dieser Gruppe werden die KDCDEF-Steueranweisungen USER erzeugt.

KC_KSET_STMT

Steht für die 4. Gruppe, die KSETS. In dieser Gruppe werden die KDCDEF-Steueranweisungen KSET erzeugt.

KC_LTAC_STMT

Steht für die 5. Gruppe, die Transaktionscodes, über die Service-Programme in Partner-Anwendungen gestartet werden. In dieser Gruppe werden die KDCDEF-Steueranweisungen LTAC erzeugt.

KC_CON_STMT

Steht für die 6. Gruppe, die Transportverbindungen zu LU6.1-Anwendungen.
In dieser Gruppe werden die KDCDEF-Steueranweisungen CON erzeugt.

KC_LSES_STMT

Steht für die 7. Gruppe, die LU6.1-Sessions. In dieser Gruppe werden die KDCDEF-Steueranweisungen LSES erzeugt.

Die Felder der Datenstrukturen müssen Sie wie folgt versorgen:

create_control_stmts

Hier geben Sie an, ob für die zu KC_INVDEF_TYPE gehörende Objektgruppe KDCDEF-Steueranweisungen erzeugt werden sollen oder nicht.

'Y' Für diese Objektgruppe sollen KDCDEF-Steueranweisungen erzeugt werden.

'N' Für diese Objektgruppe sollen keine KDCDEF-Steueranweisungen erzeugt werden. Statt 'N' kann auch das Null-Byte ('\0') angegeben werden.

file_name Name der Datei, in der die KDCDEF-Steueranweisungen abgelegt werdensollen. Der Name darf bis zu 54 Zeichen lang sein. Er muss den Konventionen für Dateinamen in dem Betriebssystem genügen, in dem die Anwendung abläuft. Auf Unix-, Linux- und Windows-Systemen kann der Dateiname absolut oder relativ angegeben werden. Eine relative Angabe bezieht sich auf das Dateiverzeichnis, in dem die Anwendung gestartet wurde.

file_mode Schreibmodus der Datei in *file_name* bzw. des Elements *elem_name.*

'C' Create:

UTM soll eine neue Datei mit dem Namen *file_name* bzw. ein neues Element mit dem Namen *elem_name* erzeugen.Auf BS2000-Systemen erzeugt der inverse KDCDEF eine SAM-Datei bzw. ein LMS-Bibliothekselement, wobei Folgendes gilt:

- Falls bereits eine Datei gleichen Namens existiert, muss es eine SAM-Datei sein. Die bereits existierende SAM-Datei wird dann überschrieben.
- Falls bereits ein Element gleichen Namens existiert und falls bei vers=C‘<version>‘ oder *HIGHEST-EXISTING oder *UPPER-LIMIT angegeben wird, dann wird ein bereits vorhandenes Element mit der angegebenen Version überschrieben.

Existiert auf Unix-, Linux- und Windows-Systemen bereits eine Datei gleichen Namens, dann wird sie überschrieben.

'E' Extend:

UTM soll die KDCDEF-Steueranweisungen an eine existierende Datei bzw. an ein bereits existierendes Element anhängen. Dabei gilt:

- Existiert die Datei mit dem Namen *file_name* nicht, legt UTM sie an.
- Wird auf BS2000-Systemen in *lib_name* eine LMS-Bibliothek angegeben, dann muss die Bibliothek bereits existieren. Dabei wird ein vorhandenes Element mit der angegebenen Version erweitert; existiert das Element mit dieser Version noch nicht, dann wird es neu angelegt.

lib_name (nur auf BS2000-Systemen)

Name der LMS-Bibliothek, in der die KDCDEF-Steueranweisungen abgelegt werden sollen. Der Name darf bis zu 54 Zeichen lang sein. Er muss den Konventionen für Dateinamen auf BS2000-Systemen genügen.

Ist der Name kürzer als die Feldlänge, dann muss er mit Leerzeichen aufgefüllt werden.

Die gleichzeitige Angabe von *file_name* und *lib_name* ist nicht erlaubt.

Wenn *lib_name* angegeben wird, dann müssen auch die Parameter *elem_name*, *vers* und *type* versorgt werden.

elem_name (nur auf BS2000-Systemen)

Name des LMS-Bibliothekselements, in das die KDCDEF-Steueranweisungen geschrieben werden sollen. Der Name darf bis zu 64 Zeichen lang sein. Ist der Name kürzer als die Feldlänge, dann muss er mit Leerzeichen aufgefüllt werden. Der Name muss den Konventionen für LMS-Elementnamen genügen.

vers (nur auf BS2000-Systemen)

Version des LMS-Bibliothekselements, in das die KDCDEF-Steueranweisungen geschrieben werden sollen. Die Version darf bis zu 24 Zeichen lang sein und muss den Konventionen für LMS-Versionsangaben genügen. Ist die Version kürzer als die Feldlänge, dann muss sie mit Leerzeichen aufgefüllt werden.

Als Version können auch die folgenden Zeichenfolgen angegeben werden:

*HIGHEST-EXISTING

Es wird in die höchste in der Bibliothek vorhandene Version des angegebenen Elements geschrieben.

*UPPER-LIMIT

Es wird in die höchst mögliche Version des angegebenen Elements geschrieben; diese Version wird von LMS durch "@" dargestellt.

*INCREMENT

Es wird für das angegebene Element eine neue Version angelegt. Die Angabe von *INCREMENT ist nur bei *file_mode*='C' erlaubt.

Diese Zeichenfolgen dürfen nicht abgekürzt werden!

type (nur auf BS2000-Systemen)

Type des LMS-Bibliothekselements, in das die KDCDEF-Steueranweisungen geschrieben werden sollen. Der Typ darf bis zu 8 Zeichen lang sein und muss den Konventionen für LMS-Typangaben genügen. Ist der Typ kürzer als die Feldlänge, dann muss er mit Leerzeichen aufgefüllt werden.

Es wird empfohlen, als *type* den LMS-Typ "S" zu verwenden.

> **i** Von KDCDEF wird nicht geprüft, ob die Angaben bei *elem_name*, *vers* oder *type* den Syntaxregeln von LMS entsprechen. Weitere Informationen zu den Syntaxregeln für den Namen von LMS-Elementen und Angabe von Version und Typ finden Sie im Handbuch „LMS SDF-Format “.

stmt_type Wenn als Maincode ein Wert ungleich KC_MC_OK zurückgegeben wird, dann enthält das Feld *stmt_type* den Index aus KC_INVDEF_TYPE, auf den sich die Fehlermeldung bezieht.

file_error_code

Wenn im Fehlerfall als Subcode KC_SC_FILE_ERROR zurückgegeben wird, dann enthält das Feld *file_error_code* den DMS-Fehlercode oder (auf BS2000-Systemen) den zugehörigen PLAM-Fehlercode.

Der Vektor *type_list* wird beim Aufruf von UTM in der Reihenfolge vom 1.Vektorelement (Index KC_DEVICE_STMT) zum letzten Vektorelement (Index KC_LSES_STMT) abgearbeitet.

Soll UTM KDCDEF-Steueranweisungen für alle Objektgruppen erzeugen, dann muss in jedem Vektorelement das Feld *create_control_stmts* mit 'Y' versorgt, der Dateiname im Feld *file_name* angegeben und der Schreibmodus der Datei im Feld *file_mode* gesetzt werden.

Sollen alle Steueranweisungen in **eine** Datei geschrieben werden, ist zu beachten, dass der Schreibmodus richtig gesetzt wird.

Bei der ersten Angabe der Datei bzw. des LMS-Bibliothekselements können Sie den Schreibmodus auf 'C' oder 'E' setzen. In den folgenden Vektorelementen muss der Schreibmodus aber auf 'E' gesetzt werden, sonst werden die zuvor erzeugten Steueranweisungen überschrieben.

Soll UTM für eine der Objektgruppen keine Steueranweisungen erzeugen, dann ist im zugehörigen Vektorelement nur *create_control_stmts*='N' oder keine Angabe zu machen.

retcode

Im Feld *retcode* liefert UTM den Returnscode des Aufrufs zurück. Neben den in Abschnitt „Returncodes" aufgelisteten Returncodes können zusätzlich folgende Returncodes auftreten:

Maincode = KC_MC_OK Der Aufruf wurde fehlerfrei bearbeitet. **Subcode:**
KC_SC_ASYN_INIT Der Auftrag wurde entgegengenommen, der inverse KDCDEF wird asynchron gestartet, sobald alle Transaktionen beendet sind, die Konfigurationsdaten verändern.

Maincode = KC_MC_DATA_INVALID Ungültige oder fehlende Angabe im Datenbereich. **Subcodes:**
KC_SC_DATA_MISSING In der Datenstruktur, die im Datenbereich übergeben wurde, sind keine Daten angegeben.
KC_SC_NO_INFO In der Datenstruktur, die im Datenbereich übergeben wurde, wurden ungültige Daten angegeben.
KC_SC_FILE_LIBRARY_MISMATCH (nur auf BS2000-Systemen) Es wurde sowohl ein Dateiname (*file_name*) als auch eine LMS-Bibliothek (*lib_name*) angegeben.
KC_SC_LMS_ELEMENT_MISSING (nur auf BS2000-Systemen) Es wurde eine LMS-Bibliothek (*lib_name*) angegeben, aber kein Element-Name (*elem_name).*
KC_SC_LMS_VERSION_MISSING (nur auf BS2000-Systemen) Es wurde eine LMS-Bibliothek (*lib_name*) angegeben, aber keine Element-Version (*vers*).
KC_SC_LMS_TYPE_MISSING (nur auf BS2000-Systemen) Es wurde eine LMS-Bibliothek (*lib_name*) angegeben, aber kein Element-Typ (*type*).
KC_SC_LMS_VERSION_MODE_MISMATCH (nur auf BS2000-Systemen) Als LMS-Version wurde *INCREMENT angegeben, aber *file_mode* ist nicht 'C'.

Maincode = KC_MC_MEMORY_INSUFF

Es steht nicht genügend von UTM intern benötigter Speicherplatz zur Verfügung.

Subcode:

KC_SC_NO_INFO

Maincode = KC_MC_REJECTED_CURR

Der Aufruf kann zur Zeit nicht bearbeitet werden.

Subcode:

KC_SC_INVDEF_RUNNING

Es läuft bereits ein inverser KDCDEF oder ein inverser KDCDEF-Lauf ist asynchron in Vorbereitung, d. h. der Auftrag kann derzeit nicht bearbeitet werden.

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen.

Subcodes:

KC_SC_NOT_GEN

Es sollen KDCDEF-Steueranweisungen für Objekte erzeugt werden, deren Typ nicht generiert wurde, z.B. USER-Anweisungen für eine Anwendung ohne Benutzerkennungen.

KC_SC_FILE_ERROR

Eine der Dateien, in die die KDCDEF-Steueranweisungen abgelegt werden sollen, kann nicht beschrieben werden. Im Feld *file_error_code* wird ein DMS-Fehlercode oder (auf BS2000-Systemen) ein PLAM-Fehlercode zurückgegeben, der Hinweise auf den aufgetretenen Fehler gibt.

KC_SC_NO_INFO

Der Pagepool zur Zwischenspeicherung der Übergabeparameter ist voll.

KC_SC_CLUSTER_CONF_INCONS

Nur für UTM-Cluster-Anwendungen:
Die laufenden Knoten-Anwendungen haben unterschiedliche Generierungen.

11.2.5 KC_DELETE_OBJECT - Objekte löschen

Mit KC_DELETE_OBJECT können Sie Objekte aus der Konfiguration löschen, die zu einem der folgenden Objekttypen gehören:

- Transportverbindungen zu entfernten LU6.1-Anwendungen (KC_CON)
- Keysets (KC_KSET)
- LU6.1-Sessions (KC_LSES)
- Transaktionscodes, über die Service-Programme in Partner-Anwendungen gestartet werden (KC_LTAC)
- LTERM-Partner zum Anschluss von Clients und Druckern (KC_LTERM)
- Clients und Drucker (KC_PTERM)
- Anwenderteilprogramme und VORGANG-Exits (KC_PROGRAM)
- Transaktionscodes und TAC-Queues (KC_TAC)
- Benutzerkennungen einschließlich ihrer Queues (KC_USER)

Detaillierte Angaben zum dynamischen Löschen von Objekten aus der Konfiguration finden Sie im Kapitel „Konfiguration dynamisch ändern".

Objekte, die nicht gelöscht werden dürfen

- LTERM-Partner, die zu einem LTERM-Pool oder Multiplexanschluss gehören,
- LTERM-Partner, die zu einer LTERM-Gruppe (Gruppen- oder Primary-LTERM) oder einem LTERM-Bündel (Master- oder Slave-LTERM) gehören,
- Druckersteuer-LTERMs,
- der LTERM-Partner KDCMSGLT, den UTM intern für den MSGTAC-Service erzeugt,
- Teilprogramme, die zu den Event-Exits START, SHUT, FORMAT oder INPUT gehören,
- Teilprogramme und VORGANG-Exits, die statisch ins Anwendungsprogramm gebunden sind,
- die Transaktionscodes KDCMSGTC, KDCSGNTC, KDCBADTC der Event-Services,
- Transaktionscodes, die einem Transportsystemzugangspunkt (BCAMAPPL) als SIGNON-TAC zugeordnet sind,
- die Dead Letter Queue KDCDLETQ,
- statisch gebundene Programme mit Event-Exits,
- das Administrationskommando KDCSHUT des Administrationsprogramms KDCADM,
- die für XATMI reservierten Transaktionscodes KDCTXCOM und KDCTXRLB,
- die Benutzerkennung KDCMSGUS, die UTM intern für den MSGTAC-Service erzeugt,
- eine Benutzerkennung, die einem Terminal für das automatische KDCSIGN oder einem UPIC- bzw. APPLI oder SOCKET-Client als Verbindungs-Benutzerkennung zugeordnet ist.

Beim Löschen von Objekten ist Folgendes zu beachten:

- Ein Teilprogramm bzw. ein VORGANG-Exit darf erst gelöscht werden, nachdem alle zugehörigen Transaktionscodes gelöscht wurden.
- Ein LTERM-Partner darf erst gelöscht werden, wenn ihm kein Client oder Drucker mehr zugeordnet ist.
- Eine Benutzerkennung darf erst gelöscht werden, wenn kein Benutzer oder Client mehr mit dieser Benutzerkennung angemeldet ist, d.h.:
  - In einer stand-alone-Anwendung mit SIGNON MULTI-SIGNON=NO darf der Benutzer nicht angemeldet sein.
  - In einer stand-alone-Anwendung mit SIGNON MULTI-SIGNON=YES darf
    - ein Benutzer mit RESTART=YES nicht angemeldet sein,
    - ein Benutzer mit RESTART=NO nicht über eine Terminal-Verbindung angemeldet sein.
  - In einer UTM-Cluster-Anwendung mit SIGNON MULTI-SIGNON=NO darf
    - ein echter Benutzer nicht angemeldet sein,
    - ein Verbindungs-Benutzer nicht an der Knoten-Anwendung angemeldet sein, an der der Administrations-Aufruf zum Löschen ausgeführt wird.
  - In einer Cluster-Anwendung mit SIGNON MULTI-SIGNON=YES darf
    - ein echter Benutzer mit RESTART=YES nicht angemeldet sein,
    - ein Verbindungs-Benutzer nicht an der Knoten-Anwendung angemeldet sein, an der der Administrations-Aufruf zum Löschen ausgeführt wird,
    - ein Benutzer mit RESTART=NO nicht über eine Terminal-Verbindung an der Knoten-Anwendung angemeldet sein, an der der Administrations-Aufruf zum Löschen ausgeführt wird.
- Ein Client/Drucker darf zum Zeitpunkt des Löschens nicht mit der Anwendung verbunden sein.
- Eine logische Verbindung für die verteilte Verarbeitung über LU6.1 darf nur gelöscht werden, wenn sie nicht aufgebaut ist.
- Eine LU6.1-Session darf nur dann gelöscht werden, wenn sie nicht aufgebaut ist und sich nicht im Zustand P (prepare to commit) befindet.

Auswirkungen des Löschens während des Anwendungslaufes

Beim Löschen sind zwei Fälle zu unterscheiden:

- das sofortige Löschen (mit *subopcode1*=KC_IMMEDIATE).
  Das sofortige Löschen einer Benutzerkennung oder eines CON-Objektes bewirkt, dass der Platz in der Objekttabelle sofort freigegeben wird und wiederverwendet werden kann. „Sofort löschen“ können Sie nur Benutzerkennungen (KC_USER) und Transportverbindungen zu LU6.1-Anwendungen (KC_CON). Sie können nach dem Löschen eine Benutzerkennung oder ein CON-Objekt mit demselben Namen neu erzeugen.
  Das sofortige Löschen ist nur in stand-alone UTM-Anwendungen möglich.
- das verzögerte Löschen (mit *subopcode1*=KC_DELAY)
  Das verzögerte Löschen hat die Bedeutung einer „dauerhaften Sperre“. Durch das verzögerte Löschen eines Objektes wird kein Platz in der Objekttabelle freigegeben. Der Name des gelöschten Objekts bleibt belegt, d.h. es kann innerhalb derselben Namensklasse kein neues Objekt mit gleichem Namen dynamisch erzeugt werden.
  In stand-alone UTM-Anwendungen ist das verzögerte Löschen von Transportverbindungen zu LU6.1-Anwendungen (KC_CON) nicht möglich.

  In UTM-Cluster-Anwendungen ist nur das verzögerte Löschen möglich.
  In UTM-Cluster-Anwendungen kann man mit einer Änderungsgenerierung Objekte löschen ohne dass die gesamte UTM-Cluster-Anwendung beendet werden muss. Um diese Änderung in allen laufenden Knoten-Anwendungen wirksam werden zu lassen, müssen die einzelnen Knoten-Anwendungen nacheinander beendet und mit der neuen Generierung gestartet werden.

  Details dazu siehe openUTM-Handbuch „Einsatz von UTM-Anwendungen“ Unterkapitel „Änderungsgenerierung im Cluster“.

Das Löschen eines Objektes kann nicht rückgängig gemacht werden.

Der inverse KDCDEF erzeugt für gelöschte Objekte keine KDCDEF-Steueranweisungen.

Welche Auswirkungen das Löschen eines Objektes auf noch nicht bearbeitete Asynchron-Aufträge, Asynchron-Nachrichten, offene Dialog-Vorgänge etc. hat, die mit dem entsprechenden Objekt verbunden sind, ist im Kapitel „Konfiguration dynamisch ändern" beschrieben.

Ablauf / Wirkungsdauer / Transaktionssicherung / Cluster

Der Aufruf unterliegt der Transaktionssicherung. Das Objekt wird erst am Ende des Teilprogrammlaufs (beim PEND) aus der Konfiguration gelöscht. Der Aufruf kann durch einen RSET-Aufruf in der gleichen Transaktion zurückgesetzt werden.

Das Löschen wirkt bei UTM-S- und UTM-F-Anwendungen über das Anwendungsende hinaus und kann nicht rückgängig gemacht werden.

In UTM-Cluster-Anwendungen (Unix-, Linux- und Windows-Systeme) gilt:
Der Aufruf wirkt Cluster-global, d.h. in allen Knoten-Anwendungen werden Objekte aus der Konfiguration gelöscht.

Versorgung der zu übergebenden Bereiche

Funktion des Aufrufs	Angabe im			
	Parameterbereich [1]	Identifikations-bereich	Selektions-bereich	Daten-bereich
Transportverbindungen zu LU6.1-Anwendungen löschen	*subopcode1*: KC_DELAY oder KC_IMMEDIATE (siehe "Auswirkungen des Löschens während des Anwendungslaufes") *obj_type*: KC_CON	Name der Partner-Anwendung, Name des Rechners, lokaler Anwendungsname	——	——
Keyset löschen	*subopcode1*: KC_DELAY *obj_type*: KC_KSET	Name des Keysets	——	——
LU6.1 Session löschen	*subopcode1*: KC_DELAY *obj_type*: KC_LSES	Lokaler Half-Session-Name	——	——
Transaktionscode löschen, über den Service-programme in Partner-Anwendungen gestartet werden	*subopcode1*: KC_DELAY *obj_type*: KC_LTAC	Name des Transaktionscodes	——	——
LTERM-Partner aus der Konfiguration löschen	*subopcode1*: KC_DELAY *obj_type*: KC_LTERM	Name des LTERM-Partners	——	——
Client oder Drucker aus der Konfiguration löschen	*subopcode1*: KC_DELAY *obj_type*: KC_PTERM	Name des Client /Druckers, Rechnername, BCAMAPPL-Name	——	——
Teilprogramm aus der Konfiguration löschen	*subopcode1*: KC_DELAY *obj_type*: KC_PROGRAM	Programmname	——	——
Transaktionscode oder TAC-Queue aus der Konfiguration löschen	*subopcode1*: KC_DELAY *obj_type*: KC_TAC	TAC-Name	——	——
Benutzerkennung einschließlich ihrer Queue aus der Konfiguration löschen	*subopcode1*: KC_DELAY oder KC_IMMEDIATE (siehe "Auswirkungen des Löschens während des Anwendungslaufes") *obj_type*: KC_USER	Benutzerkennung	——	——

[1] In allen Fällen muss im Parameterbereich der Operationscode KC_DELETE_OBJECT angegeben werden.

Versorgung der Parameter	
Parameterbereich	
Feldname	Inhalt
version	KC_ADMI_VERSION_1
retcode	KC_RC_NIL
version_data	KC_VERSION_DATA_11
opcode	KC_DELETE_OBJECT
subopcode1	KC_DELAY / KC_IMMEDIATE
obj_type	Objekttyp
obj_number	1
id_lth	Länge Objektname im Identifikationsbereich
select_lth	0
data_lth	0
Identifikationsbereich	
Objektname	
Selektionsbereich	
—	
Datenbereich	
—	

KDCADMI-Aufruf
KDCADMI (&parameter_area, &identification_area, NULL, NULL)

Rückgaben von UTM	
Parameterbereich	
Feldname	Inhalt
retcode	Returncodes

subopcode1

Im Feld *subopcode1* legen Sie die Art des Löschens fest.

KC_DELAY

wenn das Objekt als gelöscht gekennzeichnet, d.h. dauerhaft gesperrt, werden soll (verzögertes Löschen).
In stand-alone openUTM-Anwendungen ist KC_DELAY bei *obj_type*=KC_CON nicht erlaubt.

KC_IMMEDIATE

ist nur in stand-alone openUTM-Anwendungen bei *obj_type*=KC_USER und *obj_type*=KC_CON erlaubt.

KC_IMMEDIATE müssen Sie angeben, wenn eine Benutzerkennung oder eine LU6.1-Verbindung sofort gelöscht werden soll.

obj_type

Im Feld *obj_type* müssen Sie den Typ des Objektes angeben, das gelöscht werden soll. Folgende Objekttypen können Sie angeben:

KC_CON, KC_KSET, KC_LSES, KC_LTAC, KC_LTERM, KC_PROGRAM, KC_PTERM, KC_TAC (Transaktionscode einschließlich TAC-Queue) und KC_USER (Benutzerkennung einschließlich ihrer Queue)

obj_number

Pro Aufruf kann nur ein Objekt gelöscht werden, deshalb ist *obj_number*=1 zu setzen.

id_lth

Im Feld *id_lth* müssen Sie die Länge des Objektnamens angeben, den Sie im Identifikationsbereich an UTM übergeben.

Identifikationsbereich

Im Identifikationsbereich müssen Sie den Namen des Objektes übergeben, das gelöscht werden soll. Der Name des Objektes muss vollständig angegeben werden. Folgende Angaben müssen Sie machen:

bei *obj_type*=KC_CON:
in der Datenstruktur *kc_long_triple_str* der Union *kc_id_area* den Namen der Partner-Anwendung, den Namen des Rechners, auf dem sich die Anwendung befindet, und den Namen der lokalen Anwendung (BCAMAPPL-Name des CONs).

bei *obj_type*=KC_KSET:
den Namen des Keysets (*kc_name8* in der Union *kc_id_area*).

bei *obj_type*=KC_LSES:
den Namen der lokalen Half Session (*kc_name8* in der Union *kc_id_area*).

bei *obj_type*=KC_LTAC:
den Namen des Transaktionscodes, über den ferne Service-Programme gestartet werden (*kc_name8* in der Union *kc_id_area*).

bei *obj_type*=KC_PTERM:
in der Datenstruktur *kc_long_triple_str* der Union *kc_id_area* den Namen des Clients/Druckers, den Namen

des Rechners, auf dem er sich befindet, und den Namen der lokalen Anwendung (d.h. BCAMAPPL-Name des PTERMs).

bei *obj_type*=KC_PROGRAM:
den Namen des Teilprogramms (*kc_name32* in der Union *kc_id_area).*

bei *obj_type*=KC_TAC:
den Namen des Transaktionscodes oder der TAC-Queue (*kc_name8* in der Union *kc_id_area).*

bei *obj_type*=KC_USER:
den Namen der Benutzerkennung (*kc_name8* in der Union *kc_id_area).*

retcode

Im Feld *retcode* liefert UTM den Returncode des Aufrufs zurück. Neben den in Abschnitt „Returncodes" aufgelisteten Returncodes können zusätzlich folgende Returncodes auftreten. Einige dieser Returncodes können unabhängig vom angegebenen Objekttyp auftreten, andere treten nur für bestimmte Objekttypen auf.

Typunabhängige Returncodes:

Maincode = KC_MC_REJECTED Der Aufruf wurde von UTM abgewiesen. **Subcodes:**
KC_SC_INVALID_OBJECT Das angegebene Objekt existiert nicht.
KC_SC_DELETE_NOT_ALLOWED Das Objekt kann nicht gelöscht werden, es wurde bereits gelöscht oder es wurde gerade (in derselben Transaktion) erzeugt.
KC_SC_JCTL_RT_CODE_NOT_OK Nur bei UTM-Cluster-Anwendungen: Beim globalen Löschen eines Objekts ist ein UTM-interner Fehler aufgetreten. Bitte wenden Sie sich an die Systembetreuung.
KC_SC_NO_GLOB_CHANG_POSSIBLE Nur bei UTM-Cluster-Anwendungen: Keine globalen Administrationsänderungen möglich, da die Generierung der Knoten-Anwendungen zur Zeit nicht konsistent ist.
KC_SC_GLOB_CRE_DEL_LOCKED Nur bei UTM-Cluster-Anwendungen: Das Löschen eines Objekts ist zur Zeit nicht möglich, weil in einer Knoten-Anwendung das Erzeugen oder Löschen eines Objekts oder das Erzeugen, Löschen oder Aktivieren eines RSA-Schlüsselpaars noch nicht abgeschlossen ist.

Maincode = KC_MC_REJECTED_CURR

Der Aufruf kann zur Zeit nicht bearbeitet werden.

Subcode:

KC_SC_INVDEF_RUNNING

Es läuft gerade ein inverser KDCDEF, d.h. der Auftrag kann z.Zt. nicht bearbeitet werden.

Maincode = KC_MC_RECBUF_FULL

Der Puffer mit Wiederanlauf-Information ist voll. (Siehe openUTM-Handbuch „Anwendungen generieren“, KDCDEF-Steueranweisung MAX, Parameter RECBUF)

Subcode:

KC_SC_NO_INFO

Returncodes bei obj_type = KC_CON:

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen.

Subcode:

KC_SC_CONNECTED

Die angegebene LU6.1-Verbindung kann nicht gelöscht werden, weil sie z.Zt. aufgebaut ist.

Maincode = KC_MC_PAR_INVALID

Im Parameterbereich wurde ein ungültiger Wert angegeben oder ein Feld wurde nicht gesetzt.

Subcode:

KC_SC_SUBOPCODE1

Nur bei UTM-Cluster-Anwendungen:
Die angegebene LU6.1-Verbindung kann nicht gelöscht werden, Löschen mit Subcode KC_IMMEDIATE nicht erlaubt.

Returncodes bei obj_type = KC_KSET:

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen.

Subcode:

KC_SC_KSET_NOT_ADMINISTRABLE

Das Keyset KDCAPLKS kann nicht gelöscht werden.

Returncodes bei obj_type = KC_LSES:

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen.

Subcodes:

KC_SC_CONNECTED

Die LU6.1-Session kann nicht gelöscht werden, weil sie z.Zt. einer Verbindung zugeordnet ist.

KC_SC_PTC_STATE

Die Session ist im Transaktionsstatus P (prepare to commit). In diesem Zustand kann sie nicht gelöscht werden.

KC_SC_NOT_ALLOWED

Die Session ist z.Zt. belegt (nicht aktiv).

Returncodes bei obj_type = KC_LTERM:

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen.

Subcodes:

KC_SC_LTERM_IS_MASTER

Der LTERM-Partner kann nicht gelöscht werden, weil er Master eines LTERM-Bündels ist.

KC_SC_LT_DEL_GROUP_MASTER

Der LTERM-Partner kann nicht gelöscht werden, weil er Primary-LTERM einer LTERM-Gruppe ist.

KC_SC_LT_DEL_SLAVE

Der LTERM-Partner kann nicht gelöscht werden, weil er Slave eines LTERM-Bündels ist.

KC_SC_LT_DEL_ALIAS

Der LTERM-Partner kann nicht gelöscht werden, weil er Gruppen-LTERM einer LTERM-Gruppe ist.

KC_SC_REF_PTERM_NOT_DELETED

Der LTERM-Partner kann nicht gelöscht werden, weil ein Client/Drucker, der dem LTERM-Partner zugeordnet ist, noch nicht gelöscht ist.

KC_SC_LTERM_IS_CTERM

Der angegebene LTERM-Partner ist ein Druckersteuer-LTERM. Er kann nicht gelöscht werden.

KC_SC_OBJECT_TYPE_NOT_LTERM

Das angegebene Objekt kann nicht gelöscht werden, weil:

- es sich um einen LTERM-Partner handelt, der zu einem LTERM-Pool oder Multiplexanschluss gehört.
- der angegebene Name zu einem LPAP- oder OSI-LPAP-Partner gehört.

KC_SC_LTERM_NOT_ADMINISTRABLE

Der angegebene LTERM-Partner ist nicht administrierbar (z.B. der LTERM-Partner KDCMSGLT, den UTM intern für den Event-Service MSGTAC erzeugt).

Returncodes bei obj_type = KC_PROGRAM:

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen. Das Objekt konnte nicht gelöscht werden.

Subcodes:

KC_SC_REF_TAC_NOT_DELETED

Ein zum angegebenen Teilprogramm gehörender Transaktionscode ist noch nicht gelöscht.

KC_SC_PROGRAM_IS_STATIC

Das Teilprogramm kann nicht aus der Konfiguration gelöscht werden, weil es zu einem Lademodul mit Lademodus STATIC gehört.

KC_SC_PROGRAM_IS_USER_EXIT

Das angegebene Objekt ist ein Event-Exit, der mit einer KDCDEF-Steueranweisung EXIT statisch konfiguriert wurde (START-, SHUT-, FORMAT- oder INPUT-Exit).

Returncode bei obj_type = KC_PTERM:

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen.

Subcodes:

KC_SC_PTERM_CONNECTED

Der angegebene Client/Drucker kann nicht gelöscht werden, weil er z.Zt. mit der Anwendung verbunden ist.

KC_SC_OBJECT_TYPE_NOT_PTERM

Das angegebene Objekt kann nicht gelöscht werden, weil:

- es sich um einen Client handelt, der sich über einen LTERM-Pool an die Anwendung angeschlossen hat, d.h. nicht explizit konfiguriert wurde.
- auf einem BS2000-System der angegebene Name bei der KDCDEF-Generierung mit einer MUX-Anweisung (Multiplexanschluss) erzeugt wurde.

- der angegebene Name zu einem Objekt gehört, das für die verteilte Verarbeitung über OSI TP oder LU6.1 konfiguriert ist.

Returncode bei obj_type = KC_TAC:

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen.

Subcodes:

KC_SC_TAC_NOT_ADMINISTRABLE

Der angegebene Transaktionscode bzw. die angegebene Queue ist nicht administrierbar (z.B. KDCMSGTC, KDCBADTC, KDCSGNTC) bzw. darf nicht gelöscht werden (der Transaktionscode KDCSHUT und die Dead Letter Queue).

KC_SC_DELETE_NOT_ALLOWED

Der angegebene Transaktionscode darf nicht gelöscht werden (z.B. ein Transaktionscode der einem Transportzugangspunkt als SIGNON-TAC zugewiesen ist).

Returncodes bei obj_type = KC_USER (*subopcode1* = KC_DELAY oder KC_IMMEDIATE):

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen.

Subcodes:

KC_SC_USER_CONNECTED

Es ist zur Zeit ein Client/Benutzer mit der angegebenen Benutzerkennung bei der Anwendung angemeldet.

KC_SC_APPLICATION_WITHOUT_USER

Die Anwendung ist ohne Benutzerkennungen generiert.

KC_SC_USER_NOT_ADMINISTRABLE

Die Benutzerkennung ist nicht administrierbar, z.B. die Benutzerkennung KDCMSGUS, die UTM intern für den Event-Service MSGTAC erzeugt.

KC_SC_AUTO_SIGN_USER

Das Löschen der angegebenen Benutzerkennung ist nicht möglich, weil sie einem LTERM-Partner für das automatische KDCSIGN bzw. als Verbindungs-Benutzerkennung zugeordnet ist.

obj_type = KC_USER und *subopcode1* = KC_IMMEDIATE:

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen.

Subcodes:

KC_SC_ASYN_SERVICE_RUNNING

Die Benutzerkennung kann zur Zeit nicht gelöscht werden, weil unter dieser Benutzerkennung noch ein Asynchron-Vorgang läuft.

KC_SC_CLIENT_SIGNED

Das sofortige Löschen der Benutzerkennung ist zur Zeit nicht möglich, weil noch ein UPIC-Client, eine TS-Anwendung oder ein OSI TP-Partner mit dieser Benutzerkennung angemeldet ist.

KC_SC_DEADLOCK

Deadlock beim Sperren eines Benutzer-spezifischen Langzeitspeichers (ULS).

KC_SC_TIMEOUT

Timeout beim Sperren eines Benutzer-spezifischen Langzeitspeichers (ULS).

KC_SC_OWNER_IN_TA

Ein Benutzer-spezifischer Langzeitspeicher (ULS) kann nicht gesperrt werden, weil er von einer Transaktion gesperrt ist, in der einer der KDCS-Aufrufe PEND KP oder PGWT KP abgesetzt wurde.

KC_SC_PTC_STATE

Für die Benutzerkennung befindet sich eine Transaktion im Zustand PTC (Prepare To Commit).

KC_SC_BOTTLENECK

Für die Benutzerkennung sind Vorgänge gekellert und es ist zu einem Speicherengpass gekommen.

KC_SC_ALREADY_LOCKED

Der zugeordnete ULS ist von einer anderen Transaktion gesperrt.

KC_SC_NOT_ENOUGH_TASKS

Die UTM-Anwendung hat im Moment zu wenige freie Prozesse, um auf die Sperre eines von einer PTC-Transaktion gesperrten Benutzer-spezifischen Langzeitspeichers (ULS) warten zu können. Das Löschen des Benutzers später erneut versuchen.

Maincode = KC_MC_PAR_INVALID Im Parameterbereich wurde ein ungültiger Wert angegeben oder ein Feld wurde nicht gesetzt. **Subcode:**
KC_SC_SUBOPCODE1 Nur bei UTM-Cluster-Anwendungen: Löschen mit Subcode KC_IMMEDIATE ist nicht erlaubt.

11.2.6 KC_ENCRYPT - RSA-Schlüsselpaar erzeugen, löschen, auslesen

Mit KC_ENCRYPT können Sie ein neues RSA-Schlüsselpaar für Ihre Anwendung erzeugen, ein RSA-Schlüsselpaar Ihrer Anwendung durch ein neues ersetzen, RSA-Schlüsselpaare löschen und den öffentlichen Schlüssel eines RSA-Schlüsselpaares lesen.

> UTM-Anwendungen auf BS2000-Systemen unterstützen auch die Verschlüsselungen auf Verbindungen zu einigen Terminalemulationen. Auf diesen Verbindungen wird jedoch nicht das RSA-Schlüsselpaar von UTM verwendet, sondern ein von VTSU-B erzeugtes. Die Änderungen des RSA-Schlüsselpaares von UTM haben also keine Wirkung auf die Verschlüsselung über VTSU-B.

Voraussetzungen

Diese Funktion können Sie nur einsetzen, wenn die Verschlüsselungsfunktionen verfügbar sind.

Verschlüsselungsverfahren

Für eine erhöhte Sicherheit auf Verbindungen zwischen UTM-Server-Anwendungen und UPIC-Clients bietet openUTM Funktionen zur Verschlüsselung von Passwörtern und Benutzerdaten (Nachrichten) an.

Nähere Informationen zur Verschlüsselung finden Sie im openUTM-Handbuch „Konzepte und Funktionen“ und im openUTM-Handbuch „Anwendungen generieren“.

Funktionsumfang von KC_ENCRYPT

Ein RSA-Schlüsselpaar, das für eine bestimmte Verschlüsselungsebene gültig ist, wird für alle Client-Verbindungen verwendet, die diese Verschlüsselungsebene verwenden. Aus Sicherheitsgründen sollten Sie deshalb die RSA-Schlüsselpaare Ihrer UTM-Anwendung in regelmäßigen Abständen durch neue RSA-Schlüsselpaare ersetzen. Bei Encryption Level 5 wird der RSA-Schlüssel des Servers nur noch zum Signieren des öffentlichen Diffie-Hellman Schlüssels des Servers verwendet, damit der Client diesen Schlüssel eindeutig dem Server zuordnen kann. Bei diesem Verfahren ist eine längere Nutzung des RSA-Schlüssels weniger kritisch.

Für Verbindungen mit Encryption Level 5 werden ebenfalls RSA-Schlüssel mit 2048 Bits verwendet, das entspricht RSA-Schlüssln der Verschlüsselungsebene 4.

Dazu bietet KC_ENCRYPT folgende Funktionen an:

- Erzeugen eines neuen RSA-Schlüsselpaares.

  KC_ENCRYPT mit *subopcode1*=KC_CREATE_KEY veranlasst UTM ein neues RSA-Schlüsselpaar zu erzeugen. UTM setzt dieses neue RSA-Schlüsselpaar jedoch erst dann zur Verschlüsselung ein, wenn Sie es durch einen weiteren KC_ENCRYPT-Aufruf (mit *subopcode1*=KC_ACTIVATE_KEY) aktivieren.

  Ein neues Schlüsselpaar können Sie nur erzeugen, wenn das zuletzt erzeugte Schlüsselpaar der gleichen Verschlüsselungsebene bereits mit *subopcode1*=KC_ACTIVATE_KEY aktiviert ist oder mit *subopcode1* =KC_DELETE_KEY gelöscht wurde, d.h. für die Anwendung darf kein noch nicht aktiviertes Schlüsselpaar der gleichen Verschlüsselungsebene existieren.

- Löschen eines RSA-Schlüsselpaares.

  Mit KC_ENCRYPT mit *subopcode1*=KC_DELETE_KEY löschen Sie ein noch nicht aktiviertes Schlüsselpaar, mit *subopcode1* = KC_DELETE_ACTIVE_KEY löschen Sie ein aktiviertes Schlüsselpaar.

  Sie können nur aktivierte Schlüsselpaare der Verschlüsselungsebene 4 löschen. Die aktivierten Schlüsselpaare der Verschlüsselungsebene 3 werden von UTM immer benötigt.

- Aktivieren eines zuvor erzeugten RSA-Schlüsselpaares.

  KC_ENCRYPT mit *subopcode1*=KC_ACTIVATE_KEY bewirkt, dass ein derzeit benutztes RSA-Schlüsselpaar durch ein mit KC_ENCRYPT erzeugtes Paar ersetzt wird. D.h. beim nächsten Verbindungsaufbau zu einem entsprechend generierten Client wird der öffentliche Schlüssel des neuen RSA-Schlüsselpaares an den Client übertragen.

- Auslesen eines öffentlichen Schlüssels.

  Mit KC_ENCRYPT *subopcode1*=KC_READ_NEW_PUBLIC_KEY können Sie den öffentlichen Schlüssel eines noch nicht aktivierten RSA-Schlüsselpaares auslesen.
  Mit KC_ENCRYPT *subopcode1*=KC_READ_ACTIV_PUBLIC_KEY können Sie den öffentlichen Schlüssel eines aktiven RSA-Schlüsselpaares auslesen.

  Diese Funktion bietet Ihnen eine zusätzliche Möglichkeit, die Sicherheit der Daten auf der Verbindung zu erhöhen:

  Damit ein Client verifizieren kann, ob der über die Verbindung zur UTM-Anwendung empfangene öffentliche Schlüssel auch wirklich von der UTM-Anwendung stammt, sollten Sie den öffentlichen Schlüssel auslesen, über einen getrennten Weg an den Client übertragen und dort hinterlegen.

  Überträgt die UTM-Anwendung jetzt beim nächsten Verbindungsaufbau zum Client den öffentlichen Schlüssel an den Client, dann kann der Client den empfangenen öffentlichen Schlüssel mit dem bei ihm hinterlegten vergleichen.

Es empfiehlt sich also, den öffentlichen Schlüssel eines neu erzeugten RSA-Schlüsselpaares an alle betroffenen Clients zu verteilen. Betroffene Clients sind die Clients, die die Verschlüsselung der Nachrichten unterstützen.

Transaktionssicherung / Wirkungsdauer / Cluster

Das Erzeugen, Aktivieren und Löschen eines RSA-Schlüsselpaares unterliegt der Transaktionssicherung. Sie können innerhalb einer Transaktion ein neues Schlüsselpaar erzeugen und aktivieren. Ein neu erzeugter öffentlicher Schlüssel kann erst nach Abschluss der Transaktion gelesen werden.

Ein RSA-Schlüsselpaar bleibt solange aktiv, bis mit KC_ENCRYPT ein neues Schlüsselpaar erzeugt und aktiviert wird bzw. bis zur Neugenerierung der Anwendung. Bei einer Neugenerierung erzeugt UTM automatisch ein neues RSA-Schlüsselpaar, falls für den KDCDEF-Lauf die Anweisung OPTION GEN-RSA-KEYS=YES angegeben wird (Standardeinstellung).

Der Aufruf wirkt über den aktuellen Lauf der Anwendung hinaus.

Das Auslesen eines öffentlichen Schlüssels unterliegt nicht der Transaktionssicherung.

In UTM-Cluster-Anwendungen (Unix-, Linux- und Windows-Systeme) gilt:
Der Aufruf wirkt Cluster-global, d.h.

- wenn Sie auf einer Knoten-Anwendung mit der KC_ENCRYPT-Funktion ein neues Schlüsselpaar erzeugen, wird dieses Schlüsselpaar auch auf die anderen Knoten-Anwendungen verteilt, so dass alle Knoten-Anwendungen die gleichen Schlüsselpaare besitzen.
- wenn Sie auf einer Knoten-Anwendung ein vorher erzeugtes Schlüsselpaar aktivieren oder löschen, wird diese Aktion auch auf allen anderen Knoten-Anwendungen nachgezogen.

Versorgung der zu übergebenden Bereiche

Funktion des Aufrufs	Angabe im			
	Parameterbereich [1]	**Identifikations-bereich**	**Selektions-bereich**	**Datenbereich**
RSA-Schlüsselpaar erzeugen	*subopcode1*: KC_CREATE_KEY *subopcode2*: Verschlüsselungsebene	——	——	——
Nicht aktiviertes RSA-Schlüsselpaar löschen	*subopcode1*: KC_DELETE_KEY *subopcode2*: Verschlüsselungsebene	——	——	——
Aktiviertes RSA-Schlüsselpaar löschen	*subopcode1*: KC_DELETE_ACTIVE_KEY *subopcode2*: Verschlüsselungsebene	——	——	——
RSA-Schlüsselpaar aktivieren	*subopcode1*: KC_ACTIVATE_KEY *subopcode2*: Verschlüsselungsebene	——	——	——
Öffentlichen Schlüssel eines noch nicht aktivierten RSA-Schlüsselpaares auslesen	*subopcode1*: KC_READ_NEW_PUBLIC_KEY *subopcode2*: Verschlüsselungsebene	——	——	Zeiger auf einen Datenbereich, in den UTM den öffentlichen Schlüssel zurückliefern kann.
Öffentlichen Schlüssel des zur Zeit aktiven RSA-Schlüsselpaares auslesen	*subopcode1*: KC_READ_ACTIV_PUBLIC_KEY *subopcode2*: Verschlüsselungsebene	——	——	Zeiger auf einen Datenbereich, in den UTM den öffentlichen Schlüssel zurückliefern kann.

[1] In allen Fällen muss im Parameterbereich der Operationscode KC_ENCRYPT angegeben werden.

Versorgung der Parameter	
Parameterbereich	
Feldname	Inhalt
version	KC_ADMI_VERSION_1
retcode	KC_RC_NIL
version_data	KC_VERSION_DATA_11
opcode	KC_ENCRYPT
subopcode1	KC_CREATE_KEY / KC_ACTIVATE_KEY / KC_DELETE_KEY / KC_DELETE_ACTIVE_KEY / KC_READ_NEW_PUBLIC_KEY / KC_READ_ACTIV_PUBLIC_KEY
subopcode2	KC_ENC_LEV_3 / KC_ENC_LEV_4
obj_number	0
id_lth	0
select_lth	0
data_lth	Länge des Datenbereichs / 0
Identifikationsbereich	
—	
Selektionsbereich	
—	
Datenbereich	
—	

KDCADMI-Aufruf
KDCADMI (&parameter_area, NULL, NULL, NULL) oder KDCADMI (&parameter_area, NULL, NULL, &data_area)

Rückgaben von UTM	
Parameterbereich	
Feldname	Feldinhalt
retcode	Returncode
data_lth_ret	Länge der zurückgelieferten Daten / 0
Datenbereich	
Datenstruktur kc_encrypt_str / kc_encrypt_advanced_str / —	

subopcode1

Im Feld *subopcode1* müssen Sie angeben, welche Aktion UTM durchführen soll. Folgende Subopcodes können Sie angeben:

KC_CREATE_KEY

Es soll ein neues RSA-Schlüsselpaar erzeugt werden.

KC_ACTIVATE_KEY

Ein mit KC_ENCRYPT erzeugtes RSA-Schlüsselpaar soll aktiviert werden.

KC_DELETE_KEY

Ein noch nicht aktiviertes RSA-Schlüsselpaar soll gelöscht werden.

KC_DELETE_ACTIVE_KEY

Ein aktiviertes RSA-Schlüsselpaar soll gelöscht werden. Es können nur die aktivierten Schlüssel der Verschlüsselungsebene 4 gelöscht werden.
Diese Funktion ist nur erlaubt, wenn das Schlüsselpaar bis zum Löschzeitpunkt noch von keinem Objekt verwendet wurde. Diese Funktion können Sie z.B. nach einer Neugenerierung und anschließendem KDCUPD anwenden, um RSA-Schlüssel zu löschen, die in der neuen Generierung nicht mehr benötigt werden.

KC_READ_NEW_PUBLIC_KEY

Der öffentliche Schlüssel eines noch nicht aktivierten RSA-Schlüsselpaares soll ausgelesen werden.

KC_READ_ACTIV_PUBLIC_KEY

Der öffentliche Schlüssel des aktiven RSA-Schlüsselpaares soll ausgelesen werden.

subopcode2

Im Feld *subopcode2* müssen Sie angeben, auf welche Verschlüsselungsebene sich die in *subopcode1* spezifizierte Aktion bezieht:

KC_ENC_LEV_3

Die Aktion soll für RSA-Schlüssel mit Schlüssellänge 1024 Bits gelten.

KC_ENC_LEV_4

Die Aktion soll für RSA-Schlüssel mit Schlüssellänge 2048 Bits gelten.

data_lth

Im Feld *data_lth* geben Sie Folgendes an:

- bei *subopcode1*=KC_CREATE_KEY, KC_DELETE_KEY, KC_DELETE_ACTIVE_KEY oder KC_ACTIVATE_KEY:
  data_lth=0. Beim Aufruf von KDCADMI sollten Sie für *&data_area* den Nullpointer an UTM übergeben.
- bei *subopcode1*=KC_READ_NEW_PUBLIC_KEY oder KC_READ_ACTIV_PUBLIC_KEY:
  Die Länge des Datenbereichs, in dem UTM den öffentlichen Schlüssel des RSA-Schlüsselpaares zurückliefern soll. Dieser Datenbereich muss die Länge der Datenstruktur *kc_encrypt_advanced_str* besitzen.
  Beim Aufruf von KDCADMI müssen Sie den Zeiger auf den Datenbereich an UTM übergeben.

retcode

Im Feld *retcode* liefert UTM den Returncode des Aufrufs zurück. Neben den in Abschnitt „Returncodes" aufgelisteten Returncodes können zusätzlich folgende Returncodes auftreten.

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen.

Subcodes:

KC_SC_NO_ENCRYPTION

Verschlüsselung wird nicht unterstützt.

KC_SC_NEW_KEY_ALREADY_EXISTS

Bei *subopcode1*= KC_CREATE_KEY:
Es wurde bereits ein neues Schlüsselpaar für diese Verschlüsselungsebene erzeugt.

KC_SC_NO_NEW_KEY_EXISTS

Bei *subopcode1*=KC_READ_NEW_PUBLIC_KEY, KC_ACTIVATE_KEY, KC_DELETE_KEY:
Es existiert kein neuer Schlüssel für die angegebene Verschlüsselungsebene.

KC_SC_NO_ACTIV_KEY_EXISTS

Bei *subopcode1*= KC_READ_ACTIV_PUBLIC_KEY, KC_DELETE_ACTIVE_KEY:
Es existiert kein aktivierter Schlüssel für die angegebene Verschlüsselungsebene.

KC_SC_IN_USE_DEL_NOT_ALLOWED

Bei *subopcode1*=KC_DELETE_ACTIVE_KEY:

- Das Schlüsselpaar für die angegebene Verschlüsselungsebene darf nicht gelöscht werden, weil es noch von mindestens einem Objekt benötigt wird.
- Es ist nicht erlaubt, ein Schlüsselpaar der Verschlüsselungsebene 3 zu löschen (dieses Schlüsselpaar wird von UTM immer benötigt).

KC_SC_NO_GLOB_CHANG_POSSIBLE

Nur bei UTM-Cluster-Anwendungen:
Keine globalen Administrationsänderungen möglich, da die Generierung der Knoten-Anwendungen zur Zeit nicht konsistent ist.
Maßnahme: Bitte zu einem späteren Zeitpunkt wiederholen.

KC_SC_GLOB_CRE_DEL_LOCKED

Nur bei UTM-Cluster-Anwendungen:
Das Erzeugen, Löschen oder Aktivieren eines RSA-Schlüsselpaares ist zur Zeit nicht möglich, weil in einer Knoten-Anwendung das Erzeugen oder Löschen eines Objekts oder das Erzeugen, Löschen oder Aktivieren eines RSA-Schlüsselpaars noch nicht abgeschlossen ist.
Maßnahme: Bitte zu einem späteren Zeitpunkt wiederholen.

Maincode = KC_MC_RECBUF_FULL **Subcode:**
KC_SC_NO_INFO Der Puffer mit Wiederanlauf-Information ist voll. (Siehe openUTM-Handbuch „Anwendungen generieren“, KDCDEF-Steueranweisung MAX, Parameter RECBUF)

Maincode = KC_MC_REJECTED_CURR Der Aufruf kann zur Zeit nicht bearbeitet werden. **Subcode:**
KC_SC_INVDEF_RUNNING Nur bei UTM-Cluster-Anwendungen: Es läuft gerade ein inverser KDCDEF, d.h. der Auftrag kann z.Zt. nicht bearbeitet werden.

data_lth_ret

data_lth_ret enthält die Länge der Daten, die UTM im Datenbereich zurückliefert.

- bei *subopcode1*=KC_READ_NEW_PUBLIC_KEY und KC_READ_ACTIV_PUBLIC_KEY ist *data_lth_ret* != 0.
  Ist die Länge in *data_lth_ret* kleiner als der bereitgestellte Datenbereich (*data_lth*), dann ist der Inhalt des Datenbereichs nur in der Länge *data_lth_ret* definiert.
- in allen anderen Fällen ist *data_lth_ret*=0

Datenbereich

Im Fall *subopcode1*=KC_READ_NEW_PUBLIC_KEY bzw. KC_READ_ACTIV_PUBLIC_KEY liefert UTM im Datenbereich die Datenstruktur *kc_encrypt_advanced_str* mit dem öffentlichen Schlüssel der angegebenen Verschlüsselungsebene zurück. Bei KC_READ_NEW_PUBLIC_KEY wird der Schlüssel des noch nicht aktivierten RSA-Schlüsselpaares und bei KC_READ_ACTIV_PUBLIC_KEY der Schlüssel des aktiven RSA-Schlüsselpaares zurückgegeben.

Die Datenstruktur *kc_encrypt_advanced_str* ist folgendermaßen definiert:

struct kc_encrypt_advanced_str

`char buf_lth[4];`
`char en_buffer[2048];`
`char en_key_lth[4];`

Die Felder der Datenstruktur haben folgende Bedeutung:

buf_lth	gibt die verwendete Länge des Datenpuffers *en_buffer* an.
en_buffer	enthält den ausgelesenen öffentlichen Schlüssel.
en_key_lth	länge des Schlüssels (1024 oder 2048).

11.2.7 KC_GET_OBJECT - Informationen abfragen

Mit KC_GET_OBJECT können Sie Informationen über alle Objekte der Konfiguration und die Einstellung der Anwendungsparameter abfragen.

Es können verschiedene Arten von Informationen abgefragt werden. Welche Informationen UTM zurückliefern soll, können Sie über den Parameter *subopcode1* steuern.

Folgende Informationen können von UTM geliefert werden:

- Eine Liste der Namen von Objekten eines Objekttyps (*subopcode1*=KC_NAME oder KC_NAME_NEXT).
- Eigenschaften, Status- und Statistikinformationen zu den Objekten eines Objekttyps (*subopcode1* =KC_ATTRIBUTES oder KC_ATTRIBUTES_NEXT).

  Unter Eigenschaften versteht man die Parameter, die beim Konfigurieren der Objekte gesetzt wurden. UTM liefert die aktuellen Werte der Parameter zurück, d.h. Modifikationen durch die Administration werden berücksichtigt.

  Statusinformationen beschreiben den aktuellen Zustand eines Objekts, z.B. ob eine Verbindung gerade aufgebaut oder ein Benutzer gerade angemeldet ist.

  Statistikinformationen sind Zählerstände und intern gemessene Wartezeiten. UTM liefert z.B. folgende Werte zurück: die Anzahl der Nachrichten, die die Anwendung seit dem Start mit einer Partner-Anwendung bzw. einem Client ausgetauscht hat, oder die Anzahl der in einer Partner-spezifischen Message Queue zwischengespeicherten Nachrichten oder die Anzahl der Teilprogrammläufe, die über einen Transaktionscode gestartet wurden.

  Die Eigenschaften, Status- und Statistikinformationen zu einem Objekt liefert UTM im Datenbereich in der Datenstruktur des Objekttyps (siehe "Datenstrukturen zur Beschreibung der Objekteigenschaften") zurück. Liefert UTM Informationen zu mehreren Objekten zurück, dann legt UTM einen Vektor von Datenstrukturen des Objekttyps über den Datenbereich.

  Ist im Folgenden von den Eigenschaften eines Objektes die Rede, dann sind Objekteigenschaften, Status- und Statistikinformationen gemeint.
- Die aktuelle Einstellung der Anwendungsparameter(*subopcode1*=KC_APPLICATION_PAR)

  Welche Werte UTM zurückliefert, ist abhängig vom Parametertyp, den Sie in *obj_type* angeben. Sie können z.B. wählen zwischen den bei der KDCDEF-Generierung festgelegten Maximalwerten der Anwendung, den Systemparametern, den aktuellen Time-Einstellungen oder Statistikinformationen über die momentane Auslastung der Anwendung. Unter Punkt obj_type in diesem Kapitel sind die Parametertypen aufgelistet, die Sie auswählen können.
  Zu jedem Parametertyp existiert eine eigene Datenstruktur, in der UTM die angeforderten Anwendungsparameter zurückliefert. Die Datenstrukturen sind im Abschnitt "Datenstrukturen zur Beschreibung der Anwendungsparameter" beschrieben.

Steuerung der Ausgabe von Objektnamen und Objekteigenschaften

UTM liefert die Objektnamen alphabetisch sortiert zurück. Entsprechend werden auch die Eigenschaften der Objekte in der Reihenfolge der Objektnamen zurückgeliefert. Im *subopcode2* können Sie angeben, ob UTM die Namen in alphabetisch aufsteigender (KC_ASCENDING) oder absteigender (KC_DESCENDING) Reihenfolge zurückliefern soll.

Da gerade bei der Abfrage von Objekteigenschaften der Umfang der Information für alle Objekte eines Objekttyps sehr groß sein kann, sollten Sie die Menge der Informationen beschränken, die Sie anfordern. Dazu stehen Ihnen folgende Möglichkeiten zur Verfügung:

- Im Identifikationsbereich können Sie angeben, an welcher Stelle der alphabetischen Liste die Ausgabe beginnen soll. Dazu geben Sie einen beliebigen String an.

  Entspricht der String keinem Objektnamen des angegebenen Objekttyps, dann beginnt UTM die Ausgabe mit dem nächstfolgenden Objekt, d.h. mit dem alphabetisch nächstgrößeren bzw. nächstkleineren Objekt, je nachdem, was Sie in *subopcode2* angegeben haben.

  Entspricht der im Identifikationsbereich angegebene String einem Objektnamen, dann ist der Startpunkt der Ausgabe abhängig von *subopcode1*:

  - Bei *subopcode1*=KC_NAME und KC_ATTRIBUTES beginnt die Ausgabe mit diesem Objekt.
  - Bei *subopcode1*=KC_NAME_NEXT und KC_ATTRIBUTES_NEXT beginnt die Ausgabe mit dem nächstfolgenden Objekt, d.h. mit dem alphabetisch nächstgrößeren bzw. nächstkleineren Objekt, je nachdem, was Sie in *subopcode2* angegeben haben.

  Die Liste der ausgegebenen Namen bzw. Eigenschaften geht maximal bis zum letzten (*subopcode2* =KC_ASCENDING) bzw. bis zum ersten (*subopcode2*=KC_DESCENDING) Objekt in der alphabetischen Reihenfolge.

  Sollen Namen bzw. Eigenschaften der Objekte ab dem alphabetisch ersten Objekt eines Objekttyps gelesen werden, dann müssen Sie *subopcode2*=KC_ASCENDING angeben und den Identifikationsbereich mit binär null belegen.

  Sollen Namen bzw. Eigenschaften der Objekte in alphabetisch absteigender Reihenfolge ab dem letzten Objekt eines Objekttyps gelesen werden, dann müssen Sie *subop code2*=KC_DESCENDING angeben und im Identifikationsbereich den String X'FF...' übergeben.
- Im Feld *obj_number* des Parameterbereichs können Sie die maximale Anzahl der Objekte angeben, für die UTM Informationen zurückliefern soll.
- Im Selektionsbereich können Sie Selektionskriterien an UTM übergeben.

  UTM liefert dann nur Informationen zu den Objekten zurück, die diese Selektionskriterien erfüllen. Unter einem Selektionskriterium versteht man bestimmte Objekteigenschaften. So können Sie sich z.B. die Namen aller Clients/Drucker ausgeben lassen, die z.Zt. mit der Anwendung verbunden sind (*obj_type*=KC_PTERM). Unter Punkt Selektionsbereich in diesem Kapitel sind alle Selektionskriterien aufgelistet, die Sie angeben können.

  Mit der Angabe von Selektionskriterien können gezielt Objekte ausgewählt und so die Menge der Übergabedaten beschränkt werden. Die Angabe von Selektionskriterien hat jedoch Einfluss auf die Performance des Aufrufs. Insbesondere dann, wenn nur die Objektnamen abgefragt werden. UTM muss dann für jedes Objekt die Eigenschaften lesen und überprüfen, ob die entsprechende Eigenschaft mit dem Selektionskriterium übereinstimmt. D.h. ein Aufruf mit Selektionskriterien verursacht in diesem Fall wesentlich mehr Aufwand als ein Aufruf ohne Selektionskriterien.

Bei der Abfrage von Informationen ist Folgendes zu beachten

Beim Abfragen der Objektnamen bzw. der Objekteigenschaften werden auch Informationen für Objekte zurückgeliefert, die als gelöscht gekennzeichnet sind. Sie können mit Hilfe des Selektionskriteriums (*delete*='N') die Ausgabe auf die Objekte beschränken, die nicht gelöscht sind. Mit dem Selektionskriterium *delete*='Y' können Sie sich aber auch alle gelöschten Objekte des Objekttyps ausgeben lassen.

Hinweis für UTM-Cluster-Anwendungen (Unix-, Linux- und Windows-Systeme):

- In UTM-Cluster-Anwendungen werden nur Informationen über die Objekte der Knoten-Anwendung geliefert, an der der Aufruf ausgeführt wird.
- Über die Angabe KC_NO_READ_GSSBFILE und KC_NO_READ_USERFILE in *subopcode2* können Sie steuern, ob bei Folgeaufrufen für Objekte vom Typ GSSB oder USER auf die Cluster-GSSB-Datei bzw. die Cluster-User-Datei zugegriffen wird oder nicht. Damit lässt sich bei einer größeren Anzahl von Folgeaufrufen die Performance verbessern.

  Bei *subopcode2*=KC_NO_READ_GSSBFILE oder KC_NO_READ_USERFILE werden die Objekte immer in aufsteigender Reihenfolge geliefert.

  Die verbesserte Performance geht dabei einher mit einer gewissen Ungenauigkeit der Information, die in den Folgeaufrufen zurückgegeben wird. Da die Daten nicht erneut von Datei gelesen werden, sind sie möglicherweise nicht auf dem neuesten Stand.

Anwendungsmöglichkeiten

Folgende Punkte sollten Sie bei der Verwendung der Subopcodes KC_... und KC_..._NEXT berücksichtigen:

- KC_ATTRIBUTES bzw. KC_NAME sollten Sie verwenden, wenn Sie überprüfen wollen, ob bereits ein Objekt mit dem angegebenen Objektnamen existiert. Dazu geben Sie den Objektnamen im Identifikationsbereich an und setzen *obj_number*=1. Dem Returncode des Aufrufs können Sie entnehmen, ob das Objekt existiert (Sub-Returncode=KC_SC_SAME) oder nicht (Sub-Returncode=KC_SC_NEXT).
- KC_ATTRIBUTES bzw. KC_NAME können Sie als „Einstieg“ einer sukzessiven Abfrage verwenden, wenn Sie die Objektnamen ab einem bestimmten String abfragen möchten, aber nicht wissen, ob zu diesem String ein Objekt existiert oder nicht.

  Z.B. kann als Name der String 'S*bbbbbb*' (*b*=Leerzeichen) angegeben werden, wenn ab dem ersten mit „S“ beginnenden Objektnamen gelesen werden soll (solange gesichert ist, dass die Binär-Repräsentation von Leerzeichen lexikographisch kleiner ist als die von Buchstaben und Ziffern).
- KC_ATTRIBUTES und KC_NAME sind nicht geeignet für einen Folgeaufruf, bei dem Sie das zuletzt gelesene Objekt des vorherigen Aufrufs im Identifikationsbereich als Startpunkt übergeben (sukzessive Abfrage). Bei diesen Parameterwerten wird der angegebene Objektname wieder mit zurückgegeben. Bei Angabe von *obj_number*=1 und sukzessiver Abfrage würde dann immer nur das gleiche Objekt gelesen werden.

  In diesem Fall müssen Sie KC_ATTRIBUTES_NEXT bzw. KC_NAME_NEXT angeben, dann wird das folgende Objekt als erstes gelesen.

Ein Beispiel für die sukzessive Abfrage von Objekten finden Sie im Abschnitt "Beispiel für eine sukzessive Abfrage mit KC_ATTRIBUTES_NEXT".

KDCINF ("KDCINF - Informieren über Objekte und Anwendungsparameter") Der Umfang der zurückgelieferten Informationen ist bei KDCINF jedoch geringer als an der Programmschnittstelle.

Versorgung der zu übergebenden Bereiche

<table>
<tr><th rowspan="2">Funktion des Aufrufs</th><th colspan="4">Angabe im</th></tr>
<tr><th>Parameterbereich [1]</th><th>Identifikationsbereich</th><th>Selektionsbereich</th><th>Datenbereich</th></tr>
<tr><td>Namen aller Objekte eines bestimmten Objekttyps ausgeben</td><td rowspan="2">subopcode1:
KC_NAME_NEXT oder KC_NAME
subopcode2:
Ausgabe in alphabetisch auf- oder absteigender Reihenfolge
obj_type: Objekttyp
obj_number: maximale Anzahl von Objektnamen</td><td rowspan="4">Name des Objektes mit/nach dem die Ausgabe der Namen beginnen soll</td><td>——</td><td rowspan="4">(Beim Aufruf muss der Zeiger auf einen Datenbereich für die Rückgaben von UTM übergeben werden.)</td></tr>
<tr><td>Namen aller Objekte eines bestimmten Objekttyps mit bestimmten Eigenschaften ausgeben</td><td rowspan="2">Selektionskriterien, nach denen UTM die Ausgabe einschränken soll</td></tr>
<tr><td>Eigenschaften und Statistikinformationen von Objekten eines bestimmten Objekttyps mit bestimmten Eigenschaften ausgeben</td><td>subopcode1:
KC_ATTRIBUTES_NEXT oder KC_ATTRIBUTES
subopcode2:
Ausgabe in alphabetisch auf- oder absteigender Reihenfolge
obj_type: Objekttyp
obj_number: maximale Anzahl von Objekten, zu denen UTM Eigenschaften ausgeben soll</td></tr>
<tr><td>Eigenschaften und Statistikinformation von Objekten eines bestimmten Objekttyps ausgeben</td><td>subopcode1:
KC_ATTRIBUTES_NEXT oder KC_ATTRIBUTES
subopcode2:
Ausgabe in alphabetisch auf- oder absteigender Reihenfolge
obj_type: Objekttyp
obj_number: maximale Anzahl von Objekten, zu denen UTM Eigenschaften und Statistikinformationen ausgeben soll.</td><td>——</td></tr>
<tr><td>Anwendungsparameter ausgeben.</td><td>subopcode1:
KC_APPLICATION_PAR
obj_type: Parametertyp
obj_number: 0</td><td>——</td><td>——</td></tr>
</table>

[1] In allen Fällen muss im Parameterbereich der Operationscode KC_GET_OBJECT angegeben werden.

Versorgung der Parameter	
Parameterbereich	
Feldname	Feldinhalt
version	KC_ADMI_VERSION_1
retcode	KC_RC_NIL
version_data	KC_VERSION_DATA_11
opcode	KC_GET_OBJECT
subopcode1	KC_NAME_NEXT / KC_NAME / KC_ATTRIBUTES_NEXT / KC_ATTRIBUTES / KC_APPLICATION_PAR
subopcode2	KC_ASCENDING / KC_DESCENDING / KC_READ_NO_GSSBFILE / KC_READ_NO_USERFILE / binär null
obj_type	Objekttyp / Parametertyp
obj_number	Anzahl Objekte / 0
id_lth	Länge Objektname im Identifikationsbereich / 0
select_lth	Länge Daten im Selektionsbereich / 0
data_lth	Länge des Datenbereichs
Identifikationsbereich	
Objektname/ —	
Selektionsbereich	
Datenstruktur des Objekttyps mit Selektionskriterien / —	
Datenbereich	
—	

KDCADMI-Aufruf
KDCADMI (&parameter_area, &identification_area, &selection_area, &data_area) oder KDCADMI (&parameter_area, &identification_area, NULL, &data_area) oder KDCADMI (&parameter_area, NULL, NULL, &data_area)

Rückgaben von UTM	
Parameterbereich	
Feldname	Feldinhalt
retcode	Returncode
number_ret	Anzahl der Objekte
data_lth_ret	Länge der zurückgelieferten Daten
Datenbereich	
Datenstrukturen des Objekt- bzw. Parametertyps / Vektor von Objektnamen	

subopcode1

In *subopcode1* geben Sie an, welche Art der Information UTM zurückliefern soll. Folgende Werte können Sie angeben:

KC_NAME

UTM soll Namen von Objekten des Objekttyps *obj_type* zurückliefern.

Entspricht der im Identifikationsbereich angegebene String einem Objektnamen, dann soll die Ausgabe mit dem Namen dieses Objekts beginnen.

Entspricht der String im Identifikationsbereich keinem Objektnamen des angegebenen Objekttyps, dann soll UTM die Ausgabe mit dem nächstfolgenden Objekt beginnen, d.h. mit dem alphabetisch nächstgrößeren (bei *subopcode2*=KC_ASCENDING) bzw. nächstkleineren Objekt (bei *subopcode2*=KC_DESCENDING).

KC_NAME_NEXT

UTM soll Namen von Objekten des Objekttyps *obj_type* zurückliefern.

Die Ausgabe soll mit dem Objektnamen beginnen, der auf den im Identifikationsbereich angegebenen String folgt, d.h. mit dem alphabetisch nächstgrößeren bei *subopcode2* =KC_ASCENDING bzw. nächstkleineren Objekt bei *subopcode2*=KC_DESCENDING (siehe unter Punkt).

KC_ATTRIBUTES

UTM soll Eigenschaften von Objekten des Objekttyps *obj_type* zurückliefern.

Entspricht der im Identifikationsbereich angegebene String einem Objektnamen, dann soll die Ausgabe mit den Eigenschaften dieses Objekts beginnen.

Entspricht der String im Identifikationsbereich keinem Objektnamen des angegebenen Objekttyps, dann soll UTM die Ausgabe mit dem nächstfolgenden Objekt beginnen, d.h. mit dem alphabetisch nächstgrößeren (bei *subopcode2*=KC_ASCENDING) bzw. nächstkleineren Objekt (bei *subopcode2*=KC_DESCENDING).

KC_ATTRIBUTES_NEXT

UTM soll Eigenschaften von Objekten des Objekttyps *obj_type* zurückliefern.

Die Ausgabe soll mit dem Objekt beginnen, dessen Name auf den im Identifikationsbereich angegebenen String folgt, d.h. mit dem alphabetisch nächstgrößeren (bei *subopcode2* =KC_ASCENDING) bzw. nächstkleineren Objekt (bei *subopcode2*=KC_DESCENDING).

KC_APPLICATION_PAR

UTM soll die Anwendungsparameter des in *obj_type* angegebenen Parametertyps zurückliefern.

subopcode2

Die Angaben, die Sie im Feld *subopcode2* machen müssen, sind abhängig von dem in *subopcode1* gesetzten Wert.

- bei *subopcode1*=KC_APPLICATION_PAR müssen Sie *subopcode2* mit binär null (KC_NO_SUBOPCODE) versorgen.
- bei KC_NAME_NEXT, KC_NAME, KC_ATTRIBUTES_NEXT, KC_ATTRIBUTES müssen Sie in *subopcode2* einen der folgenden Werte setzen:

KC_ASCENDING,

UTM gibt die Informationen zu den Objekten in alphabetisch aufsteigender Reihenfolge der Objektnamen zurück, d.h. die alphabetisch nächstgrößeren.

KC_DESCENDING

UTM gibt die Informationen zu den Objekten in alphabetisch absteigender Reihenfolge der Objektnamen zurück, d.h. die alphabetisch nächstkleineren.

KC_READ_NO_GSSBFILE

Dieser Wert darf nur bei Folgeaufrufen in einer UTM-Cluster-Anwendung mit Objekttyp=KC_GSSB angegeben werden.
KC_READ_NO_GSSBFILE bewirkt, dass UTM nicht erneut auf die Cluster-GSSB-Datei zugreift, sondern die Daten aus dem letzten Aufruf mit KC_ASCENDING verwendet. Das Lesen von GSSBs wird dadurch performanter, siehe Hinweis unten.
UTM gibt die Informationen zu den GSSBs in aufsteigender Reihenfolge der Objektnamen zurück.

KC_READ_NO_USERFILE

Dieser Wert darf nur bei Folgeaufrufen in einer UTM-Cluster-Anwendung mit Objekttyp=KC_USER angegeben werden.
KC_READ_NO_USERFILE bewirkt, dass UTM nicht erneut auf die Cluster-User-Datei zugreift, sondern die Daten aus dem letzten Aufruf mit KC_ASCENDING verwendet. Das Lesen einer größeren Anzahl von Benutzerkennungen wird dadurch performanter, siehe Hinweis.
UTM gibt die Informationen zu den Benutzerkennungen in aufsteigender Reihenfolge der Objektnamen zurück.

Wenn Sie in UTM-Cluster-Anwendungen GSSBs oder Benutzerkennungen mit *subopcode2* =KC_ASCENDING oder *subopcode2*=KC_DESCENDING einlesen, dann werden alle Objekte von der Cluster-GSSB-Datei bzw. der Cluster-User-Datei lokal eingelesen und sortiert. Bei jedem neuerlichen Lesen von GSSBs/Benutzerkennungen mit diesem *subopcode2* werden wieder alle GSSBs (max. 30000) bzw. alle Benutzerkennungen eingelesen und sortiert.

Wenn hohe Performance gewünscht ist, dann geben Sie KC_ASCENDING nur beim ersten Aufruf an und verwenden bei allen Folgeaufrufen KC_READ_NO_GSSBFILE bzw. KC_READ_NO_USERFILE. Änderungen, die nach dem ersten Aufruf stattfinden, werden dann jedoch nicht angezeigt.

obj_type

Im Feld *obj_type* müssen Sie den Typ der Objekte bzw. der Anwendungsparameter angeben, zu denen UTM Informationen liefern soll. Welche Angaben Sie in *obj_type* machen können, ist abhängig von dem in *subopcode1* gesetzten Wert. Die erlaubten Angaben entnehmen Sie bitte der folgenden Tabelle. Die Bedeutung der Objekt-/Parametertypen ist im Abschnitt "Beschreibung der zu versorgenden Datenbereiche" (Objekt- und Parametertypen für Standard-Operationen).

Objekt-/Parametertyp	**dürfen Sie angeben bei *subopcode1* =**
Objekttyp: KC_ABSTRACT_SYNTAX KC_ACCESS_POINT KC_APPLICATION_CONTEXT KC_BCAMAPPL KC_CON KC_EDIT (nur auf BS2000-Systemen) KC_GSSB KC_KSET KC_LOAD_MODULE KC_LPAP KC_LSES KC_LTAC KC_LTERM KC_MESSAGE_MODULE KC_MUX (nur auf BS2000-Systemen) KC_OSI_ASSOCIATION KC_OSI_CON KC_OSI_LPAP KC_PROGRAM KC_PTERM KC_QUEUE KC_TAC KC_TPOOL KC_TRANSFER_SYNTAX KC_USER KC_USER_DYN1 KC_USER_DYN2 KC_USER_FIX	KC_ATTRIBUTES, KC_ATTRIBUTES_NEXT, KC_NAME, KC_NAME_NEXT
Objekttyp: KC_CHARACTER_SET (nur auf BS2000-Systemen) KC_DB_INFO KC_HTTP_DESCRIPTOR KC_PTC KC_SFUNC KC_SUBNET KC_TACCLASS	KC_ATTRIBUTES, KC_ATTRIBUTES_NEXT
Objekttyp: KC_CLUSTER_NODE	KC_ATTRIBUTES

Objekt-/Parametertyp	**dürfen Sie angeben bei *subopcode1* =**
Parametertyp: KC_CLUSTER_CURR_PAR KC_CLUSTER_PAR KC_CURR_PAR KC_DIAG_AND_ACCOUNT_PAR KC_DYN_PAR KC_MAX_PAR KC_MSG_DEST_PAR KC_PAGEPOOL KC_QUEUE_PAR KC_SIGNON KC_SYSTEM_PAR KC_TASKS_PAR KC_TIMER_PAR KC_UTMD_PAR	KC_APPLICATION_PAR

Für *obj_type*=KC_USER, KC_USER_DYN1, KC_USER_DYN2 und KC_USER_FIX ist Folgendes zu beachten:

- Für die Objekttypen KC_USER, KC_USER_DYN1, KC_USER_DYN2 und KC_USER_FIX sind die Datenstrukturen *kc_user_str*, *kc_user_fix_str*, *kc_user_dyn1_str* und *kc_user_dyn2_str* definiert.

  In stand-alone UTM-Anwendungen können die Daten zu einem Benutzer immer über die Struktur *kc_user_str* abgefragt werden.

  Die in den drei Datenstrukturen *kc_user_fix_str*, *kc_user_dyn1_str* und *kc_user_dyn2_str* enthaltenen Felder sind auch in der Datenstruktur *kc_user_str* enthalten. Die Aufteilung auf die drei Datenstrukturen wurde vorgenommen, um gezielt auf bestimmte Werte der Benutzerinformationen zugreifen zu können und auf diese Weise insbesondere die Performance beim Lesen der Benutzerinformation in UTM-Cluster-Anwendungen zu verbessern.
- Alle Daten, die die Cluster-User-Datei betreffen, liegen in der Datenstruktur *kc_user_dyn2_str*.
  Zum Lesen dieser Daten muss openUTM auf die Cluster-User-Datei zugreifen. Deswegen sollten Sie in UTM-Cluster-Anwendungen zum Lesen der Benutzerinformationen bevorzugt die neuen Objekttypen verwenden und KC_USER_DYN2 nur dann anfordern, wenn die von diesem Aufruf zurückgelieferten Daten aktuell benötigt werden.

Für *obj_type*=OSI_ASSOCIATION ist Folgendes zu beachten:

- Bei *subopcode1*=KC_NAME und KC_NAME_NEXT liefert UTM die bei der KDCDEF-Generierung festgelegten Namen der OSI TP-Associations zurück. Diese bestehen aus dem Association-Präfix, das in einer OSI-LPAP-Anweisung angegeben wurde, und einer laufenden Nummer.
  Bei diesen Werten von *subopcode1* können Sie im Identifikationsbereich einen Association-Namen angeben.
- Bei *subopcode1*=KC_ATTRIBUTES und KC_ATTRIBUTES_NEXT liefert UTM bei einem Aufruf nur die Eigenschaften von Associations zurück, die zu einer bestimmten Partner-Anwendung gehören und die entweder aufgebaut sind oder sich im Aufbau befinden. Aus diesem Grund **müssen** Sie beim Aufruf den

OSI-LPAP-Partner der Partner-Anwendung als Selektionskriterium angeben. Dazu übergeben Sie im Selektionsbereich die Datenstruktur *kc_osi_association_str* mit dem Namen des OSI-LPAP-Partners (siehe "kc_osi_association_str - Associations zu OSI TP-Partner-Anwendungen").

Die Eigenschaften einer Association sind intern nicht unter dem Association-Namen abgelegt, sondern unter einer Association-Id. Die Association-Id ordnet UTM der Association zu, solange diese aufgebaut ist. Die Zuordnung der Association-Id zu dem Namen der Association ist nicht möglich. UTM interpretiert den im Identifikationsbereich angegebenen String (Feld *kc_name8* der Union *kc_id_area*) deshalb als Association-ID. UTM liefert die Eigenschaften der aktiven Associations zu einer Partner-Anwendung nach den Association-Ids sortiert zurück. Es ist nicht möglich, die Eigenschaften zu einem Association-Namen abzufragen.

Für *obj_type*=KC_HTTP_DESCRIPTOR ist Folgendes zu beachten:

- *subopcode2* muss mit KC_ASCENDING versorgt werden.
- Der Identifikationsbereich kann verwendet werden.
- Der Selektionsbereich darf nicht verwendet werden.
- Die Ausgabe der Informationen zu den HTTP-Deskriptoren ist nicht nach Namen sortiert, sondern sie erfolgt in der Reihenfolge, in der die Anweisungen beim Empfang eines HTTP-Request ausgewertet werden um einen TAC zu bestimmen.

Für *obj_type*=KC_CHARACTER_SET ist Folgendes zu beachten:

- *subopcode2* muss mit KC_ASCENDING versorgt werden.
- Der Identifikationsbereich kann verwendet werden.
- Der Selektionsbereich darf nicht verwendet werden.
- Die Ausgabe der Informationen zu den Character Sets ist nach Namen sortiert.

Für *obj_type*=KC_SUBNET ist Folgendes zu beachten:

- *subopcode2* muss binär Null enthalten (KC_NO_SUBOPCODE).
- Der Identifikationsbereich kann verwendet werden.
- Der Selektionsbereich darf nicht angegeben werden.
- Die Ausgabe der Informationen zu den Subnetzen ist nicht nach den Subnetznamen (*mapped_name*) sortiert, sondern sie erfolgt in der Reihenfolge, in der die Anweisungen bei der Generierung angegeben wurden - getrennt nach IPv4- und IPv6-Subnetzen.

  Dies entspricht der Reihenfolge, in der die SUBNET-Einträge bei einem Verbindungsaufbau von außen ausgewertet werden.

obj_number

In *obj_number* können Sie die Anzahl der Objekte angeben, für die UTM Informationen zurückliefern soll. Folgende Angaben können Sie machen:

- bei *subopcode1*=KC_NAME, KC_NAME_NEXT, KC_ATTRIBUTES und KC_ATTRIBUTES_NEXT:
  obj_number legt die maximale Anzahl der Objekte fest, zu denen UTM Informationen zurückliefern soll. Geben Sie *obj_number*=0 an, dann liefert UTM Informationen zu so vielen Objekten zurück, wie Daten in den Datenbereich passen, bzw. weniger, wenn keine weiteren Objekte des Objekttyps mehr vorhanden sind.
  Bei *obj_type*=KC_CLUSTER_NODE ist zusätzlich Folgendes zu beachten:
  Wenn Sie eine *obj_number* > 32 angeben, setzt openUTM die *obj_number* auf 32.
- bei *subopcode1*=KC_APPLICATION_PAR müssen Sie immer *obj_number*=0 setzen.

id_lth

Welche Angabe Sie im Feld *id_lth* machen müssen, ist abhängig von der Angabe im Feld *subopcode1*:

- bei *subopcode1*=KC_NAME, KC_NAME_NEXT, KC_ATTRIBUTES und KC_ATTRIBUTES_NEXT:

  In *id_lth* müssen Sie die Länge der Datenstruktur angeben, die Sie im Identifikationsbereich an UTM übergeben.
- bei *subopcode1*=KC_APPLICATION_PAR müssen Sie immer *id_lth*=0 setzen. Der Inhalt des Identifikationsbereichs ist irrelevant.

select_lth

In *select_lth* müssen Sie einen Wert !=0 setzen, wenn Sie im Selektionsbereich Selektionskriterien an UTM übergeben wollen.

Bei *subopcode1*=KC_APPLICATION_PAR dürfen Sie keine Selektionskriterien an UTM übergeben, deshalb muss in diesem Fall immer *select_lth*=0 gesetzt werden.

Bei *subopcode1*=KC_ATTRIBUTES bzw. KC_ATTRIBUTES_NEXT und
obj_type= KC_OSI_ASSOCIATION **müssen** Sie im Selektionsbereich die Datenstruktur *kc_osi_association_str* mit dem Namen eines OSI-LPAP-Partners übergeben. In diesem Fall ist in *select_lth* die Länge der Datenstruktur *kc_osi_association_str* anzugeben.

Bei *obj_type*=KC_SUBNET und KC_HTTP_DESCRIPTOR müssen Sie immer *select_lth*=0 setzen.

data_lth

In *data_lth* müssen Sie die Länge des Datenbereichs angeben, den Sie UTM für die Rückgabe der angeforderten Informationen zur Verfügung stellen.

- Bei *subopcode1*=KC_NAME, KC_NAME_NEXT, KC_ATTRIBUTES und KC_ATTRIBUTES_NEXT gilt:
  Geben Sie *obj_number* !=0 an, dann muss der Datenbereich für die Rückgabe der angegebenen Anzahl groß genug sein. Bei *obj_number*=n (siehe .) müssen Sie für *data_lth* mindestens die Länge (n * maximale Länge des Objektnamens bzw.
  n * Länge der Datenstruktur des Objekttyps in *obj_type*) angeben.

- bei *subopcode1*=KC_APPLICATION_PAR müssen Sie mindestens die Länge der Datenstruktur des in *obj_type* gesetzten Parametertyps angeben.

Identifikationsbereich

Welche Angaben Sie im Identifikationsbereich machen müssen, ist abhängig von der Angabe im Feld *subopcode1* und dem Wert von *obj_type*:

- bei *subopcode1*=KC_NAME, KC_NAME_NEXT, KC_ATTRIBUTES und KC_ATTRIBUTES_NEXT:

  Im Identifikationsbereich müssen Sie einen String an UTM übergeben. Der String gibt an, bei welchem Objekt UTM mit der Informationsausgabe beginnen soll.

  Sie können im Identifikationsbereich auch binär null oder einen String aus nicht abdruckbaren Zeichen übergeben. UTM nimmt den angegebenen String wie er ist und sucht dazu den nächstgrößeren (bei *subopcode2*=KC_ASCENDING) bzw. nächstkleineren (bei *subopcode2*=KC_DESCENDING) Objektnamen.

  Über den Identifikationsbereich legen Sie eine Union *kc_id_area* (siehe "Beschreibung der zu versorgenden Datenbereiche" (Identifikationsbereich: identification_area)). Der String muss in dem Unionelement übergeben werden, das zu dem in *obj_type* angegebenen Objekttyp gehört.

  - Bei *obj_type*=KC_PROGRAM und KC_LOAD_MODULE
    übergeben Sie den String im Element *kc_name32.* Der Name muss linksbündig abgelegt werden, das restliche Feld muss entweder mit Leerzeichen aufgefüllt oder mit dem Null-Byte (\0) abgeschlossen werden.
    Der angegebene String muss kein Objektname sein.
  - Bei *obj_type*=KC_CON und KC_PTERM
    müssen Sie den String im Unionelement *kc_long_triple_str* übergeben. In *kc_long_triple_str* kann ein Namenstripel (Objektname, Rechnername, Name der lokalen Anwendung) angegeben werden. Der Objektname und der Name der lokalen Anwendung können bis zu 8 Zeichen lang sein, der Rechnername bis zu 64 Zeichen.
    Sie können für jeden der drei Namen einen beliebigen String angeben. Das müssen keine existierenden Namen zu sein. Es reicht auch, nur einen String für den Objektnamen anzugeben. Rechnername und den Namen der lokalen Anwendung müssen Sie nicht angeben, diese können Sie binär null setzen. Bei der Auswertung des Strings im Identifikationsbereich interpretiert UTM das Namenstripel als einen 80-stelligen Objektnamen. Der Startpunkt der Ausgabe wird entsprechend festgelegt.
  - Bei *obj_type*=KC_MUX
    müssen Sie den String im Unionelement *kc_triple_str* übergeben. In *kc_triple_str* kann ein Namenstripel (Objektname, Rechnername, Name der lokalen Anwendung) angegeben werden. Jeder Name kann bis zu 8 Zeichen lang sein. Sie können für jeden der drei Namen einen beliebigen String angeben. Das müssen keine existierenden Namen zu sein. Es reicht auch, nur einen String für den Objektnamen anzugeben. Rechnername und den Namen der lokalen Anwendung müssen Sie nicht angeben, diese können Sie binär null setzen. Bei der Auswertung des Strings im Identifikationsbereich interpretiert UTM das Namenstripel als einen 24-stelligen Objektnamen. Der Startpunkt der Ausgabe wird entsprechend festgelegt.

  - Bei *obj_type*=KC_DB_INFO
    können Sie im Unionelement *kc_name2* eine Ziffer zur Identifikation einer Datenbank angeben. Diese Ziffer repräsentiert die Datenbanken in der Reihenfolge, wie sie im KDCDEF-Lauf generiert wurden. Geben Sie einen anderen String an, wird der Aufruf abgewiesen.
  - Bei *obj_type*=KC_SFUNC
    können Sie im Unionelement *kc_name4* eine gültige Funktionstaste angeben. Geben Sie einen anderen String an, wird der Aufruf abgewiesen.

    Gültige Angaben sind:

auf BS2000-Systemen: F1 bis F20 und K1 bis K14
auf Unix-, Linux- und Windows-Systemen: F1 bis F20

Verzichten Sie auf eine Angabe im Identifikationsbereich, dann liefert UTM Informationen über alle Funktionstasten zurück.

Geben Sie eine gültige Funktionstaste an, dann beginnt UTM bei der Informationsausgabe mit dieser Funktionstaste.

- Bei *obj_type*=KC_TACCLASS
können Sie im Unionelement *kc_name2* die Werte einer existierenden TAC-Klasse angeben. Geben Sie einen anderen String an, wird der Aufruf abgewiesen.
- Bei *obj_type*=KC_OSI_ASSOCIATION
müssen Sie den String im Unionelement *kc_name8* übergeben.
Bei *subopcode1*=KC_NAME und KC_NAME_NEXT interpretiert UTM den String als Namen einer OSI TP-Association.
Bei *subopcode1*=KC_ATTRIBUTES und KC_ATTRIBUTES_NEXT interpretiert UTM den String als Association-Id. Siehe dazu die Beschreibung unter obj_type.
- Bei *obj_type*=KC_CLUSTER_NODE
müssen Sie im Identifikationsbereich LOW VALUE, HIGH VALUE oder Leerfelder übergeben. Andernfalls wird der Aufruf abgewiesen. Es wird kein bestimmter Knoten angesprochen. Wählen Sie *data_lth* so groß, dass die Information zu allen Knoten ausgegeben werden kann.
Bei allen anderen Objekttypen muss der String im Unionelement *kc_name8* übergeben werden. Der String muss linksbündig abgelegt werden, das restliche Feld sollte mit Leerzeichen aufgefüllt werden. Der angegebene String muss kein Objektname sein.
- Wird bei *obj_type*=KC_SUBNET der Identifikationsbereich verwendet, dann muss der dort angegebene Name ein Objektname sein, d.h. er muss einem generierten Subnetznamen (*mapped_name*) entsprechen, da die Information zu den Subnetzen nicht alphabetisch sortiert, sondern in der bei der Generierung angegebenen Reihenfolge abgelegt ist. Wird kein generierter *mapped_name* angegeben, dann wird als Returncode KC_MC_NO_ELT mit Subcode KC_SC_NOT_EXISTENT zurückgegeben.
- Wird bei *obj_type*=KC_HTTP_DESCRIPTOR der Identifikationsbereich verwendet, dann muss der dort angegebene Name ein Objektname sein, d.h. er muss einem generierten HTTP-Deskriptor Namen entsprechen, da die Information zu den HTTP-Deskriptoren nicht alphabetisch sortiert, sondern in der bei der Auswertung maßgeblichen Reihenfolge abgelegt ist. Wird kein generierter Name angegeben, dann wird als Returncode KC_MC_NO_ELT mit Subcode KC_SC_NOT_EXISTENT zurückgegeben.

- bei *subopcode1*=KC_APPLICATION_PAR sollte für den Identifikationsbereich der Nullpointer übergeben werden.

Selektionsbereich

Im Selektionsbereich müssen Sie bei *select_lth* `!`=0 die Datenstruktur des Objekttyps mit den Selektionskriterien an UTM übergeben. Die restlichen Felder der Datenstruktur müssen mit binär null versorgt werden.
Die Datenstrukturen sind im Abschnitt „Datenstrukturen zur Beschreibung der Objekteigenschaften" beschrieben. Der Name jeder Datenstruktur ist wie folgt aufgebaut: Zum Objekttyp „*TYP*“ gehört die Datenstruktur *„typ_* `str` “, also z.B. zu KC_LTERM gehört die Datenstruktur *kc_lterm_str*.

Bei *select_lth*=0 wird der Selektionsbereich, d.h. die Selektionskriterien, nicht ausgewertet.

Ein Selektionskriterium ist eine Objekteigenschaft. Sind Selektionskriterien angegeben, so führt UTM eine Auswahl der Objekte durch. Es werden nur Informationen zu den Objekten zurückgeliefert, die den

Selektionskriterien entsprechen. Im Folgenden ist für jeden Objekttyp aufgeführt, welche Selektionskriterien Sie angeben können.

Mögliche Selektionskriterien

- *obj_type*=KC_CON: Verbindungen zu LU6.1-Partner-Anwendungen

  Im Selektionsbereich übergeben Sie die Datenstruktur *kc_con_str* mit den Selektionskriterien. Folgende Angaben sind erlaubt:

Feldname	**Bedeutung**
connect_mode='Y'	UTM liefert Informationen über LU6.1-Verbindungen, die z.Zt. aufgebaut sind.
pronam_long	UTM liefert Informationen über LU6.1-Verbindungen zu Partner-Anwendungen, die auf einem bestimmten Rechner ablaufen. In *pronam_long* geben Sie den Namen des Rechners an.
delete	*delete*='Y': UTM liefert Informationen über LU6.1-Verbindungen, die aus der Konfiguration gelöscht wurden. *delete*='N': UTM liefert Informationen über LU6.1-Verbindungen, die nicht aus der Konfiguration gelöscht wurden.

  Sie können auch mehrere Selektionskriterien zusammen angeben, d.h. mehrere Felder zusammen setzen.

- *obj_type*=KC_LPAP: LPAP-Partner

  Im Selektionsbereich übergeben Sie die Datenstruktur *kc_lpap_str* mit den Selektionskriterien. Folgende Angaben sind erlaubt:

Feldname	**Bedeutung**
master	*master* enthält den Namen eines Master-LPAPs eines LPAP-Bündels.UTM liefert Informationen über die Slave-LPAPs dieses LPAP-Bündels,

- *obj_type*=KC_LSES: Sessions zu LU6.1-Partner-Anwendungen

  Im Selektionsbereich übergeben Sie die Datenstruktur *kc_lses_str* mit den Selektionskriterien. Folgende Angaben sind erlaubt:

Feldname	**Bedeutung**
connect_mode='Y'	UTM liefert Informationen über Sessions, für die z.Zt. eine Transportverbindung aufgebaut ist.
lpap	UTM liefert Informationen über Sessions, die einer bestimmten LU6.1-Partner-Anwendung zugeordnet sind. In *lpap* geben Sie den Namen des LPAP-Partners an, der dieser Partner-Anwendung zugeordnet ist.
delete	*delete*='Y': UTM liefert Informationen über Sessions, die aus der Konfiguration gelöscht wurden. *delete*='N': UTM liefert Informationen über Sessions, die nicht aus der Konfiguration gelöscht wurden.

Sie können auch mehrere Selektionskriterien zusammen angeben, d.h. mehrere Felder zusammen setzen.

- *obj_type*=KC_LTERM: LTERM-Partner

  Im Selektionsbereich übergeben Sie die Datenstruktur *kc_lterm_str* mit den Selektionskriterien. Folgende Angaben sind erlaubt:

Feldname	Bedeutung
master	*master* enthält den Namen eines Master-LTERM in einem LTERM-Bündel: UTM liefert Informationen über die Slave-LTERMs des LTERM-Bündels zum angegebenen Master-LTERM, *master* enthält den Namen eines Primary-LTERM in einer LTERM-Gruppe: UTM liefert Informationen über die Gruppen-LTERMs der LTERM-Gruppe zum angegebenen Primary-LTERM.
delete	*delete*='Y': UTM liefert Informationen über LTERMs, die aus der Konfiguration gelöscht wurden. *delete*='N': UTM liefert Informationen über LTERMs, die nicht aus der Konfiguration gelöscht wurden.

  Sie können auch beide Selektionskriterien zusammen angeben, d.h. beide Felder zusammen setzen.

- *obj_type*=KC_MUX: Multiplexanschlüsse (nur auf BS2000-Systemen)

  Im Selektionsbereich übergeben Sie die Datenstruktur *kc_mux_str* mit den Selektionskriterien. Folgende Angaben sind erlaubt:

Feldname	Bedeutung
connect_mode='Y'	UTM liefert Informationen über Multiplexanschlüsse, für die z.Zt. eine Transportverbindung zum Nachrichtenverteiler aufgebaut ist.
pronam	UTM liefert Informationen über Multiplexanschlüsse, die für Nachrichtenverteiler auf einem bestimmten Rechner definiert sind. In *pronam* geben Sie den Namen des Rechners an.

  Sie können auch beide Selektionskriterien zusammen angeben, d.h. beide Felder zusammen setzen.

- *obj_type*=KC_OSI_ASSOCIATION: Associations zu OSI TP-Partner-Anwendungen

  Bei *subopcode1*=KC_NAME und KC_NAME_NEXT darf kein Selektionskriterium angegeben werden.

  Bei *subopcode1*=KC_ATTRIBUTES und KC_ATTRIBUTES_NEXT **müssen** Sie das folgende Selektionskriterium an UTM übergeben (siehe dazu die Beschreibung unter Punkt obj_type). Dazu übergeben Sie im Selektionsbereich die Datenstruktur *kc_osi_association_str* mit folgender Angabe:

Feldname	Bedeutung
osi_lpap	UTM liefert Informationen über Associations, die einer bestimmten OSI TP-Partner-Anwendung zugeordnet sind. In *osi_lpap* geben Sie den Namen des OSI-LPAP-Partners an, der dieser Partner-Anwendung zugeordnet ist.

- *obj_type*=KC_OSI_LPAP: Eigenschaften von OSI TP-Partner-Anwendungen

  Im Selektionsbereich übergeben Sie die Datenstruktur *kc_osi_lpap_str* mit den Selektionskriterien. Folgende Angaben sind erlaubt:

Feldname	Bedeutung
master	*master* enthält den Namen eines Master-LPAP in einem OSI-LPAP-Bündel: UTM liefert Informationen über die Slave-LPAPs des LPAP-Bündels zum angegebenen Master-LPAP,

- *obj_type*=KC_PROGRAM: Teilprogramme

  Im Selektionsbereich übergeben Sie die Datenstruktur *kc_program_str* mit den Selektionskriterien. Folgende Angaben sind erlaubt:

Feldname	Bedeutung
load_module	UTM liefert Informationen über Teilprogramme und VORGANG-Exits, die in einem bestimmten Lademodul, Shared Object, DLL eingebunden sind. In *load_module* geben Sie den Namen des Lademoduls/Shared Objects/DLLs an.
delete	*delete*='Y': UTM liefert Informationen über Teilprogramme, die aus der Konfiguration gelöscht wurden. *delete*='N': UTM liefert Informationen über Teilprogramme, die nicht aus der Konfiguration gelöscht wurden.

  Sie können auch beide Selektionskriterien zusammen angeben, d.h. beide Felder zusammen setzen.

- *obj_type*=KC_PTERM: Clients und Drucker

  Im Selektionsbereich übergeben Sie die Datenstruktur *kc_pterm_str* mit den Selektionskriterien. Folgende Angaben sind erlaubt:

Feldname	Bedeutung
lterm	ist nur sinnvoll bei Druckern: UTM soll Informationen über die Drucker eines Druckerbündels zurückliefern. Die Drucker eines Druckerbündels sind demselben LTERM-Partner zugeordnet. In *lterm* ist der Name des LTERM-Partners anzugeben.
connect_mode='Y'	UTM liefert Informationen über Clients/Drucker, die z.Zt. mit der Anwendung verbunden sind.
pronam_long	UTM liefert Informationen über Clients/Drucker, die auf einem bestimmten Rechner ablaufen bzw. an diesem Rechner angeschlossen sind. In *pronam_long* geben Sie den Rechnernamen an.

Feldname	Bedeutung
delete	*delete*='Y': UTM liefert Informationen über Clients/Drucker, die aus der Konfiguration gelöscht wurden. *delete*='N': UTM liefert Informationen über Clients/Drucker, die nicht aus der Konfiguration gelöscht wurden.

Das Selektionskriterium *lterm* können Sie nur alleine angeben. Alle anderen Felder der Datenstruktur müssen dann mit binär null versorgt werden.
connect_mode und *pronam_long* bzw. *pronam_long* und *delete* können Sie zusammen angeben.
connect_mode und *delete* dürfen nicht zusammen gesetzt werden.

- *obj_type*=KC_USER, KC_USER_DYN1, KC_USER_DYN2, KC_USER_FIX: Benutzerkennungen

  Im Selektionsbereich übergeben Sie die Datenstruktur *kc_user_str* bzw. *kc_user_dyn1_str* mit den Selektionskriterien. Folgende Angaben sind erlaubt:

Feldname	Bedeutung
connect_mode='Y'	UTM liefert Informationen über Benutzerkennungen, mit denen z.Zt. ein Benutzer /Client bei der Anwendung angemeldet ist.
delete	*delete*='Y': UTM liefert Informationen über Benutzerkennungen, die aus der Konfiguration gelöscht wurden. *delete*='N': UTM liefert Informationen über Benutzerkennungen, die nicht aus der Konfiguration gelöscht wurden.

  Die Selektionskriterien dürfen nicht zusammen angegeben werden, d.h. pro Aufruf darf nur ein Feld gesetzt werden.

- *obj_type*=KC_LTAC oder KC_TAC: Transaktionscodes

  Im Selektionsbereich übergeben Sie die Datenstruktur *kc_ltac_str* (KC_LTAC) oder *kc_tac_str* (KC_TAC) mit den Selektionskriterien. Folgende Angaben sind erlaubt:

Feldname	Bedeutung
delete	*delete*='Y': UTM liefert Informationen über Transaktionscodes, die aus der Konfiguration gelöscht wurden. *delete*='N': UTM liefert Informationen über Transaktionscodes, die nicht aus der Konfiguration gelöscht wurden.

retcode

Im Feld *retcode* liefert UTM den Returncode des Aufrufs zurück. Neben den im Abschnitt „Returncodes" aufgelisteten Returncodes können zusätzlich folgende Returncodes auftreten.

Maincode = KC_MC_OK

Der Aufruf wurde fehlerfrei bearbeitet.

Subcodes:

KC_SC_SAME

Es wurde *subopcode1*=KC_NAME oder KC_ATTRIBUTES gesetzt und zu dem im Identifikationsbereich angegebenen Objektnamen existiert ein Objekt. Dieses Objekt wird im Datenbereich als erstes übergeben.

KC_SC_NEXT

Es wurde *subopcode1*=KC_NAME_NEXT oder KC_ATTRIBUTES_NEXT gesetzt. Oder es wurde *subopcode1*=KC_NAME oder KC_ATTRIBUTES gesetzt und zu dem im Identifikationsbereich angegebenen Objektnamen existiert jedoch kein Objekt. Das nächstgrößere bzw. nächstkleinere Objekt (abhängig von *subopcode2*) wird im Datenbereich als erstes übergeben.

Maincode = KC_MC_LAST_ELT

Der Aufruf wurde fehlerfrei bearbeitet, es wurden jedoch weniger Objekte gelesen als angefordert, da das letzte Objekt erreicht wurde.

Subcodes:

KC_SC_SAME

Es wurde *subopcode1*=KC_NAME oder KC_ATTRIBUTES gesetzt. Zu dem im Identifikationsbereich angegebenen Objektnamen existiert ein Objekt.
UTM hat Objektnamen bzw. -eigenschaften in den Datenbereich geschrieben, jedoch für weniger Objekte als in *obj_number* angefordert wurden oder (bei *obj_number*=0) weniger als in den zur Verfügung gestellten Datenbereich passen. Das letzte bzw. erste Objekt wurde vorher erreicht.

KC_SC_NEXT

Es wurde *subopcode1*=KC_NAME_NEXT oder KC_ATTRIBUTES_NEXT gesetzt. Oder es wurde *subopcode1*=KC_NAME oder KC_ATTRIBUTES gesetzt und zu dem im Identifikationsbereich angegebenen Objektnamen existiert jedoch kein Objekt. Das nächstgrößere bzw. nächstkleinere Objekt (abhängig von *subopcode2*) wird im Datenbereich als erstes übergeben.
UTM hat Objektnamen bzw. -eigenschaften in den Datenbereich geschrieben, jedoch für weniger Objekte als in *obj_number* angefordert wurden oder (bei *obj_number*=0) weniger als in den zur Verfügung gestellten Datenbereich passen. Das letzte bzw. erste Objekt wurde vorher erreicht.

Maincode = KC_MC_NO_ELT

Es wurde *subopcode1*=KC_NAME, KC_NAME_NEXT, KC_ATTRIBUTES oder KC_ATTRIBUTES_NEXT gesetzt. Es ist kein bzw. kein nächstes Element zu dem angegebenen Objektnamen vorhanden.

Subcode:

KC_SC_NO_INFO

KC_SC_NOT_EXISTENT (nur auf Unix-, Linux- und Windows-Systemen)

Bei *obj_type*=KC_SUBNET wurde der im Identifikationsbereich angegebene Objektname nicht gefunden.

Maincode = KC_MC_MEMORY_INSUFF

UTM kann die Funktion nicht ausführen, da UTM für die Bearbeitung intern mehr Speicherplatz benötigt, als zur Verfügung steht.

Subcode:

KC_SC_NO_INFO

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen, da kein Objekt des angegebenen Objekttyps existiert.

Subcode:

KC_SC_NOT_GEN

Ist *obj_type*=KC_DB_INFO, dann wurde bei der KDCDEF-Generierung keine Datenbank generiert.
Ist *obj_type*=KC_GSSB, dann existieren z.Zt. keine GSSBs (Globale Sekundärspeicherbereiche).
Ist *obj_type*=KC_MESSAGE_MODULE, dann wurde die Anwendung ohne KDCDEF-Steueranweisung MESSAGE generiert.
Ist *obj_type*=KC_UTMD_PAR, dann wurde die Anwendung ohne KDCDEF-Steueranweisung UTMD generiert.
Ist *obj_type*=KC_TACCLASS, dann wurde bei der KDCDEF-Generierung keine TAC-Klasse erzeugt.
Ist *obj_type*=KC_SUBNET, dann wurde kein IP-Subnet generiert.

KC_SC_NO_F_KEYS_GENERATED

Sie haben *obj_type*=KC_SFUNC angegeben, es wurden für die Anwendung jedoch keine Funktionstasten generiert. (Siehe openUTM-Handbuch „Anwendungen generieren“)

KC_SC_CCFG_FILE_READ_ERROR

Nur bei UTM-Cluster-Anwendungen:
Sie haben *obj_type*=KC_CLUSTER_PAR oder KC_CLUSTER_NODE angegeben, um Informationen über eine UTM-Cluster-Anwendung zu erhalten. Beim Lesen der Cluster-Konfigurationsdatei trat ein Fehler auf.

KC_SC_CCFG_INVAL_NODE_BUFF_LTH

Nur bei UTM-Cluster-Anwendungen: UTM-interner Fehler. Bitte wenden Sie sich an die Systembetreuung.
KC_SC_CCFG_FILE_LOCK_ERROR Nur bei UTM-Cluster-Anwendungen: Die Cluster-Konfigurationsdatei ist gesperrt.
KC_SC_CCFG_RT_CODE_NOT_OK Nur bei UTM-Cluster-Anwendungen: UTM-interner Fehler. Bitte wenden Sie sich an die Systembetreuung.
KC_SC_CUSF_USER_NOT_FOUND Nur bei UTM-Cluster-Anwendungen: Angegebener User existiert nicht.
KC_SC_CUSF_RT_CODE_NOT_OK Nur bei UTM-Cluster-Anwendungen: UTM-interner Fehler. Bitte wenden Sie sich an die Systembetreuung.

Maincode = KC_MC_NOT_EXISTENT Das angegebene Objekt existiert nicht. **Subcode:**
KC_SC_NO_INFO *obj_type*=KC_DB_INFO, KC_SFUNC oder KC_TACCLASS: Im Identifikationsbereich wurde keine gültige Datenbank-Id, Funktionstaste bzw. TAC-Klasse angegeben.

Maincode = KC_MC_SEL_INVALID Im Selektionsbereich wurden ungültige Daten angegeben. **Subcode:**
KC_SC_NO_INFO

number_ret

Nach einem Aufruf mit *subopcode1*=KC_NAME, KC_NAME_NEXT, KC_ATTRIBUTES oder KC_ATTRIBUTES_NEXT enthält *number_ret* die Anzahl der Objekte, zu denen UTM Informationen im Datenbereich zurückgeliefert hat.

Ist zu dem im Identifikationsbereich angegebenen String kein weiteres Objekt vorhanden, dann gibt UTM *number_ret*=0 und *data_lth_ret*=0 zurück und setzt einen entsprechenden Returncode.

Nach einem Aufruf mit *subopcode1*=KC_APPLICATION_PAR liefert UTM immer *number_ret*=0 zurück.

data_lth_ret

In *data_lth_ret* liefert UTM die Länge der Daten zurück, die UTM im Datenbereich hinterlegt hat.

Die Länge der zurückgelieferten Daten ist:

- bei *subopcode1*=KC_NAME, KC_NAME_NEXT:

  Anzahl der Objekte * Länge des zum Objekttyp gehörenden Namensfelds

- *subopcode1*=KC_ATTRIBUTES oder KC_ATTRIBUTES_NEXT:

  Anzahl der Objekte * Länge der Datenstruktur des Objekttyps

- *subopcode1*=KC_APPLICATION_PAR:

  Länge der Datenstruktur des Parametertyps

  Ist zu dem im Identifikationsbereich angegebenen String kein Objekt und auch kein weiteres Objekt vorhanden, dann gibt UTM *data_lth_ret*=0 zurück und setzt einen entsprechenden Returncode.

Datenbereich

Im Datenbereich liefert UTM die angeforderten Informationen zurück.

- *subopcode1*=KC_NAME, KC_NAME_NEXT:

  UTM liefert einen Vektor von Objektnamen zurück. In dem Vektor sind die Objektnamen alphabetisch aufsteigend (bei *subopcode2*=KC_ASCENDING) oder absteigend (bei *subopcode2*=KC_DESCENDING) geordnet.

  Die Länge der einzelnen Namen entspricht der Länge des Namensfeldes in der Datenstruktur des Objekttyps. Bei *obj_type*=KC_CON und KC_PTERM liefert UTM einen Vektor von Namensstrukturen des folgenden Formats zurück:

struct kc_long_triple_str
`char p_name[8];`
`char pronam[64];`
`char bcamappl[8];`

Bei *obj_type*=KC_MUX liefert UTM einen Vektor von Namensstrukturen des folgenden Formats zurück:

struct kc_triple_str

`char p_name[8];`
`char pronam[8];`
`char bcamappl[8];`

Für jedes Objekt werden die drei Felder der Datenstruktur versorgt mit:

p_name

Objektname, d.h. der Name der Verbindung, des Client, Druckers oder des Multiplexanschlusses

pronam

Name des Rechners, auf dem sich das Objekt befindet

bcamappl

Name der lokalen Anwendung, über den die Verbindung zu diesem Objekt aufgebaut wird.

Der Namensvektor beginnt bei *subopcode1*=KC_NAME_NEXT immer mit dem Objektnamen, der in Bezug auf den im Identifikationsbereich angegebenen String der alphabetisch nächstgrößere bzw. nächstkleinere ist, je nach Wert von *subopcode2*.
Bei *subopcode1*=KC_NAME sind zwei Fälle zu unterscheiden:

Existiert ein Objektname, der mit dem String, den Sie im Identifikationsbereich angegeben haben, übereinstimmt, dann beginnt der Namensvektor mit diesem Objektnamen. UTM liefert den Sub-Returncode KC_SC_SAME zurück.

Entspricht der im Identifikationsbereich angegebene String keinem Objektnamen, dann beginnt der Namensvektor wie bei *subopcode1*=KC_NAME_NEXT mit dem Objektnamen, der in Bezug auf den im Identifikationsbereich angegebenen String der alphabetisch nächstgrößere bzw. nächstkleinere ist. UTM liefert den Sub-Returncode KC_SC_NEXT zurück.

- *subopcode1*=KC_ATTRIBUTES oder KC_ATTRIBUTES_NEXT:

  UTM legt einen Vektor von Datenstrukturen des Objekttyps über den Datenbereich. Jede Datenstruktur enthält die Eigenschaften eines Objektes. Die Datenstrukturen liegen hintereinander und sind nach den Objektnamen alphabetisch auf- oder absteigend geordnet, je nach Wert von *subopcode2*.

  Die Datenstrukturen sind im Abschnitt „Datenstrukturen zur Beschreibung der Objekteigenschaften" beschrieben. Der Name jeder Datenstruktur ist wie folgt aufgebaut: Zum Objekttyp „*TYP*“ gehört die Datenstruktur *„typ_* `str` *“*, also z.B. zu KC_LTERM gehört die Datenstruktur *kc_lterm_str*.

  In den Datenstrukturen sind die Felder, die beim Eintragen des Objekts in die Konfiguration nicht angegeben wurden, mit den Standardwerten, Leerzeichen oder '0' besetzt. Felder, die nur in einem anderen Betriebssystem relevant sind, sind mit binär null versorgt.

  Mit welchem Objekt der Vektor beginnt, ist abhängig von *subopcode1* und von dem im Identifikationsbereich angegebenen Namen.

Bei *subopcode1*=KC_ATTRIBUTES_NEXT beginnt der Vektor mit dem Objekt, das in Bezug auf den im Identifikationsbereich angegebenen String das alphabetisch nächstgrößere bzw. nächstkleinere ist, je nach Wert von *subopcode2*.

Bei *subopcode1*=ATTRIBUTES sind zwei Fälle zu unterscheiden:

Existiert ein Objektname, der mit dem String, den Sie im Identifikationsbereich angegeben haben, übereinstimmt, dann beginnt der Vektor mit diesem Objekt. UTM liefert den Sub-Returncode KC_SC_SAME zurück.

Entspricht der String keinem Objektnamen, dann beginnt der Vektor wie bei *subopcode1* =KC_ATTRIBUTES_NEXT mit dem Objekt, das in Bezug auf den im Identifikationsbereich angegebenen String das alphabetisch nächstgrößere bzw. nächstkleinere ist. UTM liefert den Sub-Returncode KC_SC_NEXT zurück.

- *subopcode1*=KC_APPLICATION_PAR:

  UTM legt die Datenstruktur des in *obj_type* angegebenen Parametertyps über den Datenbereich. In der Datenstruktur liefert UTM die angeforderten Anwendungsparameter zurück.

  Die Datenstrukturen sind im Abschnitt „Datenstrukturen zur Beschreibung der Anwendungsparameter" beschrieben. Der Name jeder Datenstruktur ist wie folgt aufgebaut: Zum Parametertyp „*TYP*“ gehört die Datenstruktur „*typ_* `str`“, also z.B. zu KC_MAX_PAR gehört die Datenstruktur *kc_max_par_str*.

Beispiel für eine sukzessive Abfrage mit KC_ATTRIBUTES_NEXT

Aufgabe
Es soll die gesamte Information zu Benutzerkennungen gelesen werden, deren Namen mit „S“ beginnen. Es wird hier vorausgesetzt, dass solche Benutzerkennungen existieren.

Lösung

Erster KC_GET_OBJECT-Aufruf:
(Es wird vorausgesetzt, dass bei diesem Aufruf *n* Objekte gefunden werden, d.h. *n_ret=n* ist) Rückgaben von UTM:

Angaben im Parameterbereich:
version=KC_ADMI_VERSION_1 retcode=KC_RC_NIL version_data=KC_VERSION_DATA_11 opcode=KC_GET_OBJECT subopcode1=KC_ATTRIBUTES subopcode2=KC_ASCENDING obj_type=KC_USER obj_number=n id_lth=8 select_lth=0 data_lth=n * sizeof(struct kc_user_str)
Angaben im Identifikationsbereich: 'S *bbbbbbb* ' or 'S\0' (*b* = blank, \0 = Null-Byte in C)
Angaben im Selektionsbereich: keine
Angaben im Datenbereich: keine

Rückgaben von UTM:

Rückgaben im Parameterbereich:
retcode= KC_MC_OK with subcode KC_SC_SAME or KC_SC_NEXT number_ret=n_ret data_lth_ret=n_ret*sizeof(struct kc_user_str)
Rückgaben im Datenbereich: n_ret * Datenstruktur *kc_user_str* mit den Eigenschaften der Benutzerkennungen

Beginnt die zuletzt ausgegebene Benutzerkennung noch mit „S“, dann muss ein weiterer Aufruf erfolgen.

Zweiter KC_GET_OBJECT-Aufruf:
(Angaben, die sich von denen beim ersten Aufruf unterscheiden, sind unterstrichen)

Angaben im Parameterbereich :
version=KC_ADMI_VERSION_1 retcode=KC_RC_NIL version_data=KC_VERSION_DATA_11 opcode=KC_GET_OBJECT subopcode1=KC_ATTRIBUTES_NEXT subopcode2=KC_ASCENDING obj_type=KC_USER obj_number=n id_lth=8 select_lth=0 data_lth=n * sizeof(struct kc_user_str)
Angaben im Identifikationsbereich : Name der Benutzerkennung, die UTM beim ersten Aufruf als letzte zurückgeliefert hat
Angaben im Selektionsbereich : keine
Angaben im Datenbereich : keine

Rückgaben von UTM:

Rückgaben im Parameterbereich:
retcode=KC_MC_OK with subcode KC_SC_NEXT [1] number_ret=n_ret (<= n) data_lth_ret=n_ret * sizeof(struct kc_user_str)
Rückgaben im Datenbereich: n_ret * Datenstruktur *kc_user_str* mit den Daten der Benutzerkennungen

[1] Es können auch die Returncodes KC_MC_LAST_ELT (falls weniger als n Objekte gefunden wurden) und KC_MC_NO_ELT (falls kein weiteres Objekt gefunden wurde) auftreten.

Der zweite Aufruf wird dann solange wiederholt, bis alle Benutzerkennungen mit „S“ gelesen sind. Das erkennen Sie durch Auswertung der Rückgabedaten. D.h. beginnt der Name der Benutzerkennung, die UTM als letzte zurückliefert, mit „S“, dann muss der Aufruf nochmal wiederholt werden. Beginnt er nicht mit „S“ oder ist beim letzten Aufruf *number_ret* != *obj_number*, dann sind bereits alle Benutzerkennungen mit „S“ gelesen.

11.2.8 KC_LOCK_MGMT - Sperren in UTM-Cluster-Anwendungen aufheben

Nur auf Unix-, Linux- und Windows-Systemen.

Mit KC_LOCK_MGMT können Sie:

- Alle oder einen einzelnen Benutzer, die/der an einer abnormal beendeten Knoten-Anwendung angemeldet sind /ist, abmelden (KDCOFF). Cluster-weit gültige Vorgangsdaten dieser Benutzer gehen dabei verloren.

  Für diese Funktion verwenden Sie die Subopcodes KC_SIGNOFF_ALL und KC_SIGNOFF_SINGLE.
- Für alle oder einen einzelnen Benutzer, die/der einen an eine beendete Knoten-Anwendung gebundenen Vorgang haben/hat, diesen Vorgang zum abnormalen Beenden kennzeichnen und somit ein erneutes Anmelden der Benutzer/des Benutzers an einer anderen Knoten-Anwendung ermöglichen. Der gebundene Vorgang wird beim nächsten Start der Knoten-Anwendung, an die er gebunden ist, abnormal beendet.

  Für diese Funktion verwenden Sie die Subopcodes KC_ABORT_BOUND_SERVICE, KC_ABORT_ALL_BOUND_SERVICES und KC_ABORT_PTC_SERVICE.
- Eine Sperre der Cluster-User-Datei bei nicht ordnungsgemäß beendetem KDCDEF-Lauf wieder aufheben (Subopcode KC_UNLOCK_USF).

Wirkungsdauer / Transaktionssicherung / Cluster

Durch den Aufruf wird die Cluster-User-Datei dauerhaft geändert. Die Änderung wird sofort wirksam und kann durch Rücksetzen der Transaktion nicht rückgängig gemacht werden.

Diese Funktion steht nur für UTM-Cluster-Anwendungen zur Verfügung.

Versorgung der Parameter	
Parameterbereich	
Feldname	Feldinhalt
version	KC_ADMI_VERSION_1
retcode	KC_RC_NIL
version_data	KC_VERSION_DATA_11
opcode	KC_LOCK_MGMT
subopcode1	KC_ABORT_ALL_BOUND_SERVICES / KC_ABORT_BOUND_SERVICE / KC_ABORT_PTC_SERVICE / KC_SIGNOFF_ALL / KC_SIGNOFF_SINGLE / KC_UNLOCK_USF
id_lth	0
select_lth	0
data_lth	Länge der Datenstruktur / 0
Identifikationsbereich	
—	

Selektionsbereich
—
Datenbereich
Datenstruktur / 0

KDCADMI-Aufruf
KDCADMI (&parameter_area, NULL, NULL, &data_area)

Rückgaben von UTM	
Parameterbereich	
Feldname	Inhalt
retcode	Returncodes

subopcode1

Im Feld *subopcode1* geben Sie an, welche Aktion openUTM durchführen soll. Sie können folgende Subopcodes angeben:

KC_ABORT_ALL_BOUND_SERVICES

Kennzeichnet alle Vorgänge, die an eine beendete Knoten-Anwendung gebunden sind, zum abnormalen Beenden. Damit können sich die Benutzer an andere Knoten-Anwendungen anmelden (KDCSIGN). Die gebundenen Vorgänge werden beim nächsten Start der Knoten-Anwendung, an die sie gebunden sind, abnormal beendet.

KC_ABORT_BOUND_SERVICE

Kennzeichnet einen Vorgang eines Benutzers, der an eine beendete Knoten-Anwendung gebunden ist, zum abnormalen Beenden. Damit kann sich der Benutzer an eine andere Knoten-Anwendung anmelden (KDCSIGN). Der gebundene Vorgang wird beim nächsten Start der Knoten-Anwendung, an die er gebunden ist, abnormal beendet.

KC_ABORT_PTC_SERVICE

Kennzeichnet einen Vorgang eines Benutzers, der an eine beendete Knoten-Anwendung gebunden ist und eine Transaktion im Zustand PTC besitzt, zum abnormalen Beenden. Damit kann sich der Benutzer an eine andere Knoten-Anwendung anmelden (KDCSIGN). Der gebundene Vorgang wird beim nächsten Start der Knoten-Anwendung, an die er gebunden ist, abnormal beendet.

KC_SIGNOFF_ALL

Alle User, die an einer abnormal beendeten Knoten-Anwendung angemeldet sind, abmelden, damit sich diese an einer anderen Knoten-Anwendung anmelden können. Cluster-weit gültige Vorgangsdaten dieser Benutzer gehen dabei verloren.

KC_SIGNOFF_SINGLE

Einen einzelnen User, der an einer abnormal beendeten Knoten-Anwendung angemeldet ist, abmelden, damit dieser sich an einer anderen Knoten-Anwendung anmelden kann. Cluster-weit gültige Vorgangsdaten dieses Benutzers gehen dabei verloren.

KC_UNLOCK_USF

Hebt die Sperre in der Cluster-User-Datei wieder auf, nachdem ein KDCDEF-Lauf nicht ordnungsgemäß beendet wurde Der Aufruf mit Subopcode KC_UNLOCK_USF ist nur dann erforderlich, wenn ein KDCDEF-Lauf abnormal beendet wurde und ein nachfolgender KDCDEF-Lauf die Meldung K516 mit Fehler-Code 8 ausgibt.

data_lth

Im Feld *data_lth* geben Sie die Länge der Datenstruktur im Datenbereich oder 0 an.

Datenbereich

Im Datenbereich müssen Sie bei allen *subopcode1* außer KC_UNLOCK_USF die Datenstruktur *kc_lock_mgmt_str* angeben.

Die Datenstruktur *kc_lock_mgmt_str* ist folgendermaßen definiert:

struct kc_lock_mgmt_str
`char mg_name[8];` `char mg_node[4];`

Die Felder der Datenstruktur haben die folgende Bedeutung:

mg_name

- Bei *subopcode1*=KC_SIGNOFF_SINGLE:
  Name des Benutzers, der abgemeldet werden soll.
- Bei *subopcode1*=KC_ABORT_BOUND_SERVICE:
  Name des Benutzers, dessen an eine beendete Knoten-Anwendung gebundener Vorgang zum abnormalen Beenden gekennzeichnet werden sollen.
- Bei *subopcode1*=KC_ABORT_PTC_SERVICE:
  Name des Benutzers, dessen an eine beendete Knoten-Anwendung gebundener Vorgang im Zustand PTC zum abnormalen Beenden gekennzeichnet werden sollen.
- Bei anderem *subopcode1*: irrelevant

Die Knotennummer müssen Sie nicht angeben. Diese ermittelt openUTM.

mg_node

- Bei *subopcode1*=KC_SIGNOFF_ALL:
  Nummer des Knotens, von dem alle Benutzer abgemeldet werden sollen.
- Bei *subopcode1*=KC_ABORT_ALL_BOUND_SERVICES:
  Nummer des Knotens, der beendet wurde. Alle an diesen Knoten gebundenen Vorgänge sollen zum abnormalen Beenden gekennzeichnet werden.

- Bei anderem *subopcode1*: irrelevant

retcode

Im Feld *retcode* liefert openUTM den Returncode des Aufrufs zurück. Neben den im Abschnitt „Returncodes" aufgelisteten Returncodes können zusätzlich folgende Returncodes auftreten.

Maincode = KC_MC_REJECTED Der Aufruf wurde von UTM abgewiesen. **Subcodes:**
KC_SC_CUSF_TRANSIENT_ERROR (nur auf BS2000-Systemen) Bei jedem *subopcode1*: Temporärer Fehler beim Zugriff auf die Cluster-User-Datei; bitte den Aufruf wiederholen.
KC_SC_CUSF_RT_CODE_NOT_OK Bei jedem *subopcode1*: UTM-interner Fehler. Bitte wenden Sie sich an die Systembetreuung.
KC_SC_CUSF_INVALID_STATE Bei *subopcode1*= KC_SIGNOFF_ALL/KC_ABORT_ALL_BOUND_SERVICES: Die angegebene Knoten-Anwendung wurde noch nie gestartet oder läuft momentan. Der Aufruf kann nur in den Knoten-Stati FAIL oder ABTERM ausgeführt werden. Bei *subopcode1*= KC_SIGNOFF_SINGLE: Die Knoten-Anwendung, an die der angegebene Benutzer angemeldet ist, läuft momentan. Bei *subopcode1*= KC_ABORT_BOUND_SERVICE/KC_ABORT_PTC_SERVICE: Die Knoten-Anwendung, an die der Vorgang des angegebenen Benutzers gebunden ist, läuft momentan.
KC_SC_CUSF_USER_HAS_NO_BND_SRV Bei *subopcode1*=KC_ABORT_BOUND_SERVICE: Der User hat keinen gebundenen Vorgang.
KC_SC_CUSF_USER_HAS_NO_PTC Bei *subopcode1*=KC_ABORT_PTC_SERVICE: Der User hat keinen an einen Knoten gebundenen Vorgang mit einer Transaktion im Zustand PTC.
KC_SC_CUSF_USER_HAS_PTC Bei *subopcode1*=KC_ABORT_BOUND_SERVICE: Der User hat einen an einen Knoten gebundenen Vorgang mit einer Transaktion im Zustand PTC.
KC_SC_CUSF_USER_NOT_FOUND Bei *subopcode1*=KC_SIGNOFF_SINGLE/KC_ABORT_BOUND_SERVICE /KC_ABORT_PTC_SERVICE: Der User wurde nicht gefunden.
KC_SC_CUSF_USER_NOT_SIGNED

Bei *subopcode1*=KC_SIGNOFF_SINGLE: Der User ist an keinem Knoten angemeldet.
KC_SC_DATA_MISSING *mg_name* ist nicht binär null und *subopcode1*=KC_SIGNOFF_ALL, KC_ABORT_BOUND_SERVICE oder KC_ABORT_ALL_BOUND_SERVICES.
KC_SC_NOT_NULL *mg_node* ist nicht binär null und *subopcode1*=KC_SIGNOFF_SINGLE, KC_ABORT_BOUND_SERVICE oder KC_ABORT_PTC_SERVICE.

11.2.9 KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern

Mit KC_MODIFY_OBJECT können Sie Anwendungsparameter und Objekteigenschaften ändern sowie Aktionen für die Objekte der Anwendung durchführen. Folgende Änderungen können Sie durchführen:

Aktionen für die Objekte der Anwendung

- Verbindungen zu Clients, Druckern, Partner-Anwendungen auf- bzw. abbauen
- den automatischen Verbindungsaufbau zu Clients, Druckern, Partner-Anwendungen veranlassen
- Clients, Drucker, Partner-Anwendungen, Benutzerkennungen einschließlich ihrer Queues, Transaktionscodes und TAC-Queues sperren und wieder zulassen
- die Zuordnung zwischen Client/Drucker und LTERM-Partner ändern
- das Passwort für eine Benutzerkennung ändern
- Keys in Keysets ändern
- den Timer für die Überwachung von Leerlaufzuständen einer Session ändern oder die Überwachung ausschalten
- den UTM-BCAM-Trace Objekt- und Benutzer-spezifisch ein- und ausschalten
- Lademodule bzw. Shared Objects/DLLs eines Anwendungsprogramms austauschen
- die Master-LTERMs zweier LTERM-Bündel austauschen oder das LTERM in eine LTERM-Gruppe aufnehmen
- das Speichern von Asynchron-Nachrichten in der Dead Letter Queue (TAC-Queue KDCDLETQ) festlegen
- Lademodule auf BS2000-Systemen , die in Common Memory Pools geladen sind, für den Austausch mit KC_CHANGE_APPLICATION vormerken
- die maximale Anzahl der Clients auf BS2000-Systemen, die gleichzeitig über einen Multiplexanschluss mit der Anwendung verbunden sein können, ändern
- Rechnernamen und Filebase-Namen einer Knoten-Anwendung ändern
- Datenbank-Benutzernamen und -Passwort ändern

Aktionen für die Anwendungsparameter

- die Timer der Anwendung ändern
- Statistikdaten zurücksetzen
- Maximalwerte der Anwendung ändern
- Diagnosefunktionen ein- und ausschalten (z.B. BCAM-Trace)
- die Anzahl der Prozesse (TASKS) festlegen, die für die Anwendung arbeiten sollen
- die maximale Anzahl der Prozesse festlegen, die gleichzeitig Asynchron-Aufträge oder Vorgänge mit blockierenden Funktionsaufrufen (z.B. KDCS-Aufruf PGWT) bearbeiten dürfen
- Timer für die gegenseitige Überwachung der Knoten-Anwendungen ändern
- In UTM-Cluster-Anwendungen die Statistikwerte für die Belegung des Cluster Pagepools zurücksetzen

Übergabe der neuen Objekteigenschaften und Anwendungsparameterwerte

Zur Übergabe der neuen Objekteigenschaften bzw. Anwendungsparameter stehen in der Header-Datei *kcadminc.h* Datenstrukturen zur Verfügung. Für jeden Objekttyp und für jeden Parametertyp gibt es eine eigene Datenstruktur. Der Name der Datenstruktur entspricht dem Namen des Objekttyps/Parametertyps (in Kleinbuchstaben) mit dem

Suffix „_str“ (*objecttyp_str*, *parametertyp_str*). In der folgenden Beschreibung ist angegeben, in welchen Feldern der Datenstrukturen Sie die neuen Eigenschaften übergeben müssen. Die vollständige Beschreibung der Datenstrukturen finden Sie im Abschnitt „Datenstrukturen zur Informationsübergabe".

Beim Ändern von Objekteigenschaften oder Parametern des Anwendungsprogramms ist Folgendes zu beachten

Bei der Änderung von Objekteigenschaften können Sie mit einem KC_MODIFY_OBJECT-Aufruf nur die Eigenschaften eines Objektes ändern.
Im Identifikationsbereich müssen Sie dabei den Namen des Objektes vollständig angeben, so dass UTM das Objekt eindeutig identifizieren kann.
Objektnamen sind nicht modifizierbar.

Bei der Änderung von Anwendungsparametern können Sie innerhalb eines Aufrufs alle Parameter ändern, die zu demselben Parametertyp gehören, d.h. in einer Datenstruktur enthalten sind.

Die bei einem KC_MODIFY_OBJECT-Aufruf angegebenen transaktionalen Änderungen werden entweder alle zusammen durchgeführt oder keine der Änderungen wird durchgeführt. Für Änderungen, die nicht der Transaktionssicherung unterliegen, gilt dies nicht.

Wirkungsdauer / Transaktionssicherung / Cluster

Der Zeitpunkt, zu dem eine Änderung wirksam wird, und ihre Wirkungsdauer sind abhängig von der Art der Änderung. Von der Art der Änderung hängt es auch ab, ob die Änderung der Transaktionssicherung unterliegt oder nicht.

In einer UTM-Cluster-Anwendung gilt (Unix-, Linux- und Windows-Systeme):
Der Aufruf kann sowohl Aktionen anstoßen, die Cluster-global wirken als auch solche, die Knoten-lokal wirken. Global wirkende Aktionen werden für jede Knoten-Anwendung der UTM-Cluster-Anwendung wirksam, unabhängig davon, ob eine Knoten-Anwendung gerade aktiv ist oder nicht. Lokal wirkende Aktionen wirken sich nur auf die Knoten-Anwendung aus, an der der Aufruf durchgeführt wird. Abhängig vom Objekt, wirken alle Parameter eines Objekts global oder alle lokal oder auch gemischt global/lokal. Die Änderung kann über den aktuellen Lauf der Cluster-Anwendung hinaus wirken oder nur für den aktuellen Lauf. Modifikationen, die Auswirkungen auf die UTM-Konfiguration haben, wirken immer Cluster-global, um die Generierung konsistent zu halten. Diese globale Wirksamkeit ist in der Spalte des Operationscodes KC_MODIFY_OBJECT mit „G“ gekennzeichnet. Bei einer Kennzeichnung, in der kein „G“ enthalten ist, ist die Wirkung in einer UTM-Cluster-Anwendung Knoten-lokal.
Eine genaue Beschreibung des Wirkungsbereichs der einzelnen Parameter jedes Objekts ist bei der Beschreibung der Datenstrukturen zu finden.

Die folgenden Änderungstypen können auftreten:

IR/GIR

> Die Änderung unterliegt nicht der Transaktionssicherung. Sie wirkt sofort (**I**mmediate) und nur für den aktuellen Lauf der Anwendung/der UTM-Cluster-Anwendung (**R**un). Ein in der gleichen Transaktion nachfolgender RSET-Aufruf setzt die Änderung nicht zurück.

ID/GID

> Die Änderung unterliegt nicht der Transaktionssicherung. Sie wirkt sofort (**I**mmediate) und, unabhängig von der Generierungsvariante (UTM-S oder UTM-F), über den aktuellen Lauf der Anwendung/der UTM-Cluster-Anwendung hinaus (**D**urable). Ein in der gleichen Transaktion nachfolgender RSET-Aufruf setzt die Änderung nicht zurück.

PR/GPR

Die Änderung unterliegt der Transaktionssicherung.
Sie wird erst beim Transaktionsende wirksam (**P**end). Die Änderung wirkt nur für den aktuellen Lauf der Anwendung/der UTM-Cluster-Anwendung (**R**un). Sie ist durch einen RSET-Aufruf in der gleichen Transaktion rücksetzbar.

P/GP

Die Änderung unterliegt der Transaktionssicherung.
Sie wird erst beim Transaktionsende wirksam (**P**end). Die Dauerhaftigkeit der Änderung ist abhängig von der Generierungsvariante der Anwendung. Bei UTM-F wirkt die Änderung nur für den aktuellen Anwendungslauf, bei UTM-S über den aktuellen Anwendungslauf hinaus. Sie ist durch einen RSET-Aufruf in der gleichen Transaktion rücksetzbar.

PD/GPD

Die Änderung unterliegt der Transaktionssicherung.
Sie wird erst beim Transaktionsende wirksam (**P**end).
Die Änderung wirkt, unabhängig von der Generierungsvariante, über den aktuellen Lauf der Anwendung /der UTM-Cluster-Anwendung hinaus (**D**urable). Sie ist durch einen RSET-Aufruf in der gleichen Transaktion rücksetzbar.

A/GA

Es wird ein Auftrag (**A**ction) erzeugt um die gewünschte Änderung (z.B. Verbindungsaufbau/-abbau oder Anwendungsprogrammaustausch) zu bewirken. Wann der Auftrag bearbeitet wird, hängt von der Auslastung der Anwendung ab. Ob der Auftrag erfolgreich ausgeführt wurde oder nicht, können Sie nur durch eine spätere Informationsabfrage (z.B. mit KC_GET_OBJECT) erfahren. Der Auftrag ist nicht rücksetzbar.

Hinweis zur Wirkungsdauer in UTM-Cluster-Anwendungen (Unix-, Linux- und Windows-Systeme):

- Ist die Änderung nicht generierbar, so gilt die administrative Änderung auch nach dem Start einer Knoten-Anwendung mit neuer Generierung weiter, aber längstens bis Ende des Laufs der UTM-Cluster-Anwendung. Der Lauf einer UTM-Cluster-Anwendung beginnt mit dem Start der ersten Knoten-Anwendung und endet mit dem Beenden der letzten Knoten-Anwendung.
- Ist die Änderung auch generierbar, so gilt ab dem Start einer Knoten-Anwendung mit neuer Generierung der Generierungswert, nicht der administrativ geänderte Wert.

In der Beschreibung der möglichen Modifikationen unter Punkt "Datenbereich" ist angegeben, zu welchem Änderungstyp die einzelnen Änderungen gehören. Es werden die oben stehenden Abkürzungen verwendet.

Einen Teil der Modifikationen können Sie auch mit den Administrationskommandos durchführen. Welche Kommandos das sind, können Sie der Beschreibung unter Punkt "Datenbereich" entnehmen.

Versorgung der zu übergebenden Bereiche

Funktion des Aufrufs	Angabe im			
	Parameterbereich [1]	**Identifikationsbereich**	**Selektionsbereich**	**Datenbereich**
Eigenschaften eines Objektes ändern	*obj_type*: Objekttyp	Name des Objektes	——	Datenstruktur des Objekttyps mit den neuen Eigenschaftswerten
Anwendungsparameter ändern	*obj_type*: Parametertyp	——	——	Datenstruktur des Parametertyps mit den neuen Parameterwerten

[1] In allen Fällen muss im Parameterbereich der Operationscode KC_MODIFY_OBJECT angegeben werden.

Versorgung der Parameter	
Parameterbereich	
Feldname	Inhalt
version	KC_ADMI_VERSION_1
retcode	KC_RC_NIL
version_data	KC_VERSION_DATA_11
opcode	KC_MODIFY_OBJECT
subopcode1	KC_NO_SUBOPCODE / KC_IMMEDIATE / KC_DELAY
obj_type	Objekttyp / Parametertyp
obj_number	1 / 0
id_lth	Länge Objektname im Identifikationsbereich / 0
select_lth	0
data_lth	Länge der Datenstruktur im Datenbereich
Identifikationsbereich	
Objektname / —	
Selektionsbereich	
—	
Datenbereich	
Datenstruktur des Objekttyps bzw. Parametertyps / —	

KDCADMI-Aufruf
KDCADMI (&parameter_area, &identification_area, NULL, &data_area) oder KDCADMI (&parameter_area, NULL, NULL, &data_area) oder KDCADMI (&parameter_area, NULL, NULL, NULL)

Rückgaben von UTM	
Parameterbereich	
Feldname	Inhalt
retcode	Returncodes

subopcode1

Bei *obj_type* = KC_DB_INFO müssen Sie im Feld *subopcode1* KC_IMMEDIATE angeben, wenn die Änderung des Datenbank-Passworts sofort wirksam werden soll. Bei KC_DELAY wird die Änderung des Datenbank-Passworts erst beim nächsten Start der Anwendung wirksam. Eine Änderung des Datenbank-Benutzernamens wird immer erst beim nächsten Start der Anwendung wirksam.

Bei allen anderen Werten für *obj_type* müssen Sie in *subopcode1* KC_NO_SUBOPCODE angeben.

obj_type

Im Feld *obj_type* müssen Sie den Typ des Objektes, dessen Eigenschaften geändert werden sollen, bzw. den Typ der Anwendungsparameter, die geändert werden sollen, angeben. Folgende Angaben sind erlaubt:

Objekttypen:

- KC_CLUSTER_NODE
  (nur möglich in einer UTM-Cluster-Anwendung) geben Sie an, wenn Sie Rechnernamen und/oder Filebase-Namen einer Knoten-Anwendung ändern wollen.
  KC_CLUSTER_NODE müssen Sie z.B. angeben, wenn Sie einer Reserve-Knoten-Anwendung tatsächliche Werte für den Rechnernamen des Knotens und den Basisnamen der KDCFILE der Knoten-Anwendung zuordnen möchten (siehe openUTM-Handbuch „Anwendungen generieren“ und openUTM-Handbuch „Einsatz von UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen“).
- KC_DB_INFO
  geben Sie an, wenn Sie für eine XA-Datenbank das Datenbank-Passwort und/oder den Datenbank-Benutzernamen ändern wollen.
- KC_KSET
  geben Sie an, wenn Sie Keys in einem Keyset ändern wollen.
- KC_LOAD_MODULE
  geben Sie an, wenn Sie Lademodule einer UTM-Anwendung auf BS2000-Systemen bzw. Shared Objects /DLLs einer UTM-Anwendung auf Unix-, Linux- oder Windows-Systemen austauschen wollen, also eine andere Version eines Lademoduls/Shared Objects/DLLs ins Anwendungsprogramm laden wollen.
- KC_LPAP
  geben Sie an, wenn Sie eine Aktion für einen LPAP-Partner der Anwendung durchführen, d.h. die logischen Eigenschaften einer LU6.1-Partner-Anwendung ändern wollen.

- KC_LSES
  geben Sie an, wenn Sie die Eigenschaften einer Session mit einer LU6.1-Partner-Anwendung ändern wollen.
- KC_LTAC
  geben Sie an, wenn Sie die lokalen Eigenschaften eines fernen Services ändern wollen, d.h. die Eigenschaften eines LTACs.
- KC_LTERM
  geben Sie an, wenn Sie die Eigenschaften eines LTERM-Partners ändern wollen.
- KC_MUX (nur auf BS2000-Systemen)
  geben Sie an, wenn Sie die Eigenschaften eines Multiplexanschlusses ändern wollen.
- KC_OSI_CON
  geben Sie an, wenn Sie die Eigenschaften der Verbindungen zu einer OSI TP-Partner-Anwendung ändern wollen.
- KC_OSI_LPAP
  geben Sie an, wenn Sie eine Aktion für einen OSI-LPAP-Partner durchführen, d.h. die logischen Eigenschaften einer OSI TP-Partner-Anwendung ändern wollen.
- KC_PTERM
  geben Sie an, wenn Sie Aktionen für Terminals, Drucker, Client-Anwendungen oder TS-Anwendungen durchführen wollen.
- KC_TAC
  geben Sie an, wenn Sie die Eigenschaften eines Transaktionscodes, der einem lokalen Service zugeordnet ist, oder einer TAC-Queue ändern wollen.
- KC_TACCLASS
  geben Sie an, wenn Sie die maximale Anzahl der Prozesse ändern wollen, die gleichzeitig Aufträge für eine bestimmte TAC-Klasse bearbeiten dürfen.
- KC_TPOOL
  geben Sie an, wenn Sie die Eigenschaften der LTERM-Partner oder die Anzahl der aktiven LTERM-Partner eines LTERM-Pools ändern wollen.
- KC_USER
  geben Sie an, wenn Sie die Eigenschaften einer Benutzerkennung oder deren Queue ändern wollen.

Parametertypen

- KC_CLUSTER_CURR_PAR
  geben Sie an, wenn Sie in einer UTM-Cluster-Anwendung die Statistikwerte des Cluster Pagepools zurücksetzen möchten.
- KC_CLUSTER_PAR
  geben Sie an, wenn Sie für eine UTM-Cluster-Anwendung
  - die Parameter ändern wollen, die die Verfügbarkeitsprüfung der einzelnen Knoten-Anwendungen untereinander regeln.
  - die Parameter ändern wollen, die den Zugriff der Knoten-Anwendungen auf die Cluster-Konfigurationsdatei und das Cluster-Administrations-Journal regeln.
- KC_CURR_PAR
  geben Sie an, wenn Sie Anwendungs-spezifische Statistikwerte zurücksetzen wollen.

- KC_DIAG_AND_ACCOUNT_PAR
  geben Sie an, wenn Sie Diagnosefunktionen ein- oder ausschalten oder die Einstellungen des UTM-Accounting ändern wollen.
- KC_MAX_PAR
  geben Sie an, wenn Sie Maximalwerte (MAX-Parameter) der Anwendung ändern oder die Datenlieferung an openSM2 ein- oder ausschalten wollen.
- KC_TASKS_PAR
  geben Sie an, wenn Sie Werte ändern wollen, die sich auf die Prozessanzahl der Anwendung beziehen, d. h. Gesamtzahl der Prozesse, Maximalzahl der Prozesse für die Bearbeitung von Asynchron-Aufträge usw.
- KC_TIMER_PAR
  geben Sie an, wenn Sie die Einstellung von Timern ändern wollen.

Unter Punkt Datenbereich wird für jeden Objekttyp und Parametertyp beschrieben, welche Modifikationen möglich sind.

obj_number

Welche Angabe Sie im Feld *obj_number* machen müssen, ist abhängig von der Angabe im Feld *obj_type*:

- *obj_number*=1 geben Sie an, wenn Sie in *obj_type* einen Objekttyp angeben (Ausnahme: KC_TACCLASS, siehe unten).
- *obj_number*=0 geben Sie an, wenn Sie in *obj_type* einen Parametertyp angeben oder wenn Sie bei *obj_type* = KC_TACCLASS Werte für für alle TAC- Klassen zurücksetzen möchten.

id_lth

Welche Angabe Sie im Feld *id_lth* machen müssen, ist abhängig von der Angabe im Feld *obj_type*:

- geben Sie in *obj_type* einen Objekttyp an, dann müssen Sie in *id_lth* die Länge der Datenstruktur angeben, die Sie im Identifikationsbereich an UTM übergeben.Ausnahme: Bei *obj_type* = KC_DB_INFO und KC_TACCLASS müssen Sie *id_lth*=2 angeben.
- geben Sie in *obj_type* einen Parametertyp an, dann müssen Sie *id_lth*=0 setzen.

data_lth

Im Feld *data_lth* geben Sie die Länge der Datenstruktur an, die Sie im Datenbereich an UTM übergeben.

data_lth=0 ist nicht erlaubt.

Identifikationsbereich

Im Identifikationsbereich übergeben Sie den Namen des Objektes, dessen Eigenschaften Sie ändern wollen. Das heißt:

- geben Sie in *obj_type* einen Objekttyp an, dann müssen Sie im Identifikationsbereich den vollständigen Namen des Objektes an UTM übergeben
  Ausnahmen:
  - Bei *obj_type* = KC_TACCLASS und Rücksetzen von Werten für für alle TAC-Klassen müssen Sie binär null eintragen.
  - Bei *obj_type* = KC_DB_INFO müssen Sie eine Ziffer zur Identifikation einer Datenbank angeben. Diese Ziffer repräsentiert die Datenbanken in der Reihenfolge, wie sie im KDCDEF-Lauf generiert wurden und an der Administrationsschnittstelle bei KC_GET_OBJECT zurückgegeben werden.
    Welche Angaben Sie machen müssen, ist unter Punkt abhängig vom Objekttyp beschrieben. Geben Sie

in *obj_type* einen Parametertyp an, dann müssen Sie keinen Identifikationsbereich an UTM übergeben. Angaben im Identifikationsbereich werden von UTM ignoriert.

Datenbereich

Im Datenbereich übergeben Sie die Datenstruktur des in *obj_type* angegebenen Objekt- bzw. Parametertyps. Für jeden einzelnen Objekt- bzw. Parametertyp steht eine eigene Datenstruktur zur Verfügung, die Sie über den Datenbereich legen müssen. In der Datenstruktur müssen Sie die neuen Eigenschafts- bzw. Parameterwerte an UTM übergeben. Die restlichen Felder der Datenstruktur, d.h. die Felder der Eigenschaften bzw. Parameterwerte, die Sie nicht verändern wollen oder dürfen, müssen Sie vor dem Aufruf mit binär null versorgen.

In openUTM auf Unix- und Linux-Systemen müssen bei *obj_type* = KC_LOAD_MODULE nicht immer Daten im Datenbereich übergeben werden, da zum Austausch von Shared Objects ohne Versionsangabe der Name des Shared Objects im Identifikationsbereich ausreichend ist.

In den folgenden Tabellen im Abschnitt "obj_type=KC_CLUSTER_NODE" sind die erlaubten Modifikationen abhängig vom Objekttyp/Parametertyp beschrieben. Sie können der Beschreibung entnehmen, welche Eigenschaften/Parameter Sie ändern können und wie die Felder zu versorgen sind. Die gesamten Datenstrukturen sind im Abschnitt „Datenstrukturen zur Informationsübergabe" beschrieben.

retcode

Im Feld *retcode* liefert UTM den Returncode des Aufrufs zurück, siehe „Returncodes".

11.2.9.1 obj_type=KC_CLUSTER_NODE

Die Änderungen beziehen sich auf eine Knoten-Anwendung einer UTM-Cluster-Anwendung (Unix-, Linux- und Windows-Systeme).

Im Identifikationsbereich müssen Sie die Cluster-interne Nummer (Index des Eintrags dieses Knotens bei KC_GET_OBJECT für Objekt KC_CLUSTER_NODE) der Knoten-Anwendung angeben (Feld *kc_name2* der Union *kc_id_area*). Im Datenbereich müssen Sie die Datenstruktur *kc_cluster_node_str* mit den neuen Eigenschaftswerten übergeben. Es können nur Knoten geändert werden, die nicht aktiv sind.

In der Datenstruktur *kc_cluster_node_str* geben Sie Folgendes an:

Feldname	Bedeutung
hostname_long	*hostname_long* enthält den primären Rechnernamen des Knotens, auf dem diese Knoten-Anwendung abläuft. *hostname_long* kann bis zu 64 Zeichen lang sein.

filebase	Basisname der KDCFILE, der Benutzer-Protokolldatei und der System-Protokolldatei SYSLOG der Knoten-Anwendung. Beim Start der Knoten-Anwendung werden die UTM-Systemdateien unter dem hier angegebenen Namen erwartet. Diese Dateistruktur muss von allen Knoten-Anwendungen aus zugreifbar sein.
	Der Name wird in dem Element *filebase* vom Typ *kc_file_base* übergeben:
	struct kc_file_base
	`char length[2];`
	`char fb_name[42];`

	fb_name: Basisname
	length: Länge des Basisnamens
	Beachten SIe bei der Änderung des Basisnamens einer Knoten-Anwendung Folgendes: • Die Basisnamen der einzelnen Knoten-Anwendungen einer UTM-Cluster-Anwendung müssen sich voneinander unterscheiden. • Geben Sie das Dateiverzeichnis an, das die UTM-Systemdateien der Knoten-Anwendung enthält. Der hier angegebene Name muss aus Sicht aller Knoten das gleiche Dateiverzeichnis bezeichnen. Er kann bis zu 27 Zeichen lang sein.
virtual_host_long	übernimmt für UTM-Cluster-Anwendungen die Funktion des Parameters HOSTNAME der Generierungsanweisung MAX. Den Parameter MAX HOSTNAME dürfen Sie in UTM-Cluster-Anwendungen nicht angeben.
	Durch die Angabe von *virtual_host_long* kann die Absenderadresse für Netzverbindungen spezifiziert werden, die von dieser Knoten-Anwendung aus aufgebaut werden.

Wirkungsdauer / Transaktionssicherung: Typ GID ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

Die Wirkung ist dauerhaft. Die Information wird in der Cluster-Konfigurationsdatei abgelegt. Die Änderung wird sofort wirksam und kann durch Rücksetzen der Transaktion nicht rückgängig gemacht werden.

11.2.9.2 obj_type=KC_DB_INFO

Die Änderungen beziehen sich auf eine Datenbank.

Im Identifikationsbereich müssen Sie eine Ziffer zur Identifikation einer Datenbank angeben (Feld *kc_name2* der Union *kc_id_area*). Diese Ziffer repräsentiert die Datenbanken in der Reihenfolge, wie sie im KDCDEF-Lauf generiert wurden und an der Administrationsschnittstelle bei KC_GET_OBJECT zurückgegeben werden.

Im Datenbereich müssen Sie die Datenstruktur *kc_db_info_str* mit den neuen Eigenschaftswerten übergeben.

Mögliche Modifikation

Für eine XA-Datenbank können Sie das Datenbank-Passwort und den Datenbank-Benutzernamen ändern.

In der Datenstruktur *kc_db_info_str* geben Sie Folgendes an:

Feldname	Bedeutung
db_userid	Im Feld *db_userid* geben Sie den neuen Benutzernamen für dieses Datenbanksystem an. Die Änderung wird erst beim nächsten Start der UTM-Anwendung wirksam.
db_password	Im Feld *db_password* geben Sie das neue Passwort für dieses Datenbanksystem an. Die Änderung wird abhängig von der Angabe in *subcode1* entweder sofort oder erst beim nächsten Start der UTM-Anwendung wirksam, siehe Punkt "subopcode1" im Abschnitt "KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern".

Wirkungsdauer / Transaktionssicherung: Typ GPD ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

11.2.9.3 obj_type=KC_KSET

Die Änderungen beziehen sich auf die Keys (Key-/Zugangscodes) eines Keysets.

Im Identifikationsbereich müssen Sie den Namen des Keysets angeben (Feld *kc_name* der Union *kc_id_area*). Im Datenbereich müssen Sie die Datenstruktur *kc_kset_str* mit den neuen Eigenschaftswerten übergeben.

Mögliche Modifikation

Mit Ausnahme des MASTER-Keysets können Sie einen oder mehrere Keys eines Keysets ändern. Das Keyset muss in der Konfiguration der Anwendung existieren.

In der Datenstruktur *kc_kset_str* geben Sie Folgendes an:

Feldname	**Bedeutung**	
keys[4000]	Ein Key- oder Zugangscode ist eine ganze Zahl zwischen 1 und dem Wert KEYVALUE, der bei der KDCDEF-Generierung in der MAX-Anweisung festgelegt wurde. *keys* besteht aus 4000 Feldelementen *keys*[0] bis *keys*[3999]. Der Inhalt der Feldelemente ist folgendermaßen zu interpretieren:	
	keys[0]=	'0': Der Key- / Zugangscode 1 gehört nicht zu diesem Keyset. '1': Der Key- / Zugangscode 1 gehört zu diesem Keyset.
	keys[n]=	'0': Der Key- / Zugangscode n+1 gehört nicht zu diesem Keyset. '1': Der Key- / Zugangscode n+1 gehört zu diesem Keyset. Ist n+1 größer als KEYVALUE, darf '1' nicht angegeben werden.
	keys[3999]=	'0': Der Key- / Zugangscode 4000 gehört nicht zu diesem Keyset. '1': Der Key- / Zugangscode 4000 gehört zu diesem Keyset.

Wirkungsdauer / Transaktionssicherung: Typ GPD ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

11.2.9.4 obj_type=KC_LOAD_MODULE

Die Aktion bezieht sich auf ein Lademodul (BS2000-Systeme) bzw. auf ein Shared Object/DLL (Unix-, Linux- und Windows-Systeme).

Im Identifikationsbereich müssen Sie den Namen des Lademoduls/Shared Objects an UTM übergeben (Feld *kc_name32* der Union *kc_id_area*).

Im Datenbereich müssen Sie die Datenstruktur *kc_load_module_str* übergeben.

Mögliche Modifikation

Sie können ein Lademodul, Shared Object oder DLL des Anwendungsprogramms austauschen bzw. Lademodule im Common Memory Pool (BS2000-Systeme) zum Austausch vormerken.

Das angegebene Lademodul, Shared Object oder DLL muss in der Konfiguration der Anwendung existieren, d.h. mit KDCDEF statisch generiert sein.

In der Datenstruktur *kc_load_module_str* geben Sie Folgendes an:

Feldname	Bedeutung
version[24]	In *version* übergeben Sie die Version des Lademoduls bzw. Shared Objects, das geladen werden soll. *Folgendes gilt nur für BS2000-Systemen:* In UTM-Anwendungen auf BS2000-Systemen müssen Sie die zu ladende Version des Lademoduls immer angeben. Für Lademodule, die mit LOAD-MODE=STARTUP generiert sind, dürfen die Versionsnummern von altem und neuem Lademodul übereinstimmen. Für Lademodule, die mit LOAD-MODE=ONCALL generiert sind oder die ganz oder teilweise in einem Common Memory Pool liegen, muss sich die neue Versionsnummer von der alten Versionsnummer unterscheiden. Sie können als Version auch *HIGHEST-EXISTING angeben. UTM ermittelt dann die höchste in der Bibliothek vorhandene Version und lädt diese. In diesem Fall gibt UTM nach erfolgreichem Aufruf in dem Feld *version* die ermittelte höchste Elementversion zurück. Ist ein Lademodul mit LOAD-MODE=POOL, (POOL,STARTUP) oder (POOL,ONCALL) sowie mit Version *HIGHEST-EXISITING generiert, dann darf bei *version* **nur** *HIGHEST-EXISTING angegeben werden. Ein solcher Modul kann nur über einen Anwendungsaustausch neu geladen werden; dabei wird für einen so generierten Modul immer die höchste verfügbare Version geladen. Wird im Feld *version* der String *UPPER-LIMIT angegeben, dann ersetzt UTM diesen Wert in der Ausgabe durch "@". Beim Anstoßen des Austausches muss in der Bibliothek, die dem Lademodul bei der KDCDEF-Generierung zugeordnet wurde (siehe auch *lib* in *kc_load_module_str,* Abschnitt "kc_load_module_str - Lademodule (BS2000-Systeme) bzw. Shared Objects/DLLs (Unix-, Linux- und Windows-Systeme)"), ein Element mit dem im Identifikationsbereich angegebenen Namen und der in *version* angegebenen Version vorhanden sein. In UTM-Cluster-Anwendungen gilt dies für jede Knoten-Anwendung. Ist in der Programmbibliothek kein solches Lademodul vorhanden, dann wird der Administrationsaufruf abgewiesen und das bisher geladene Lademodul bleibt geladen. Zusätzlich wird die Meldung K234 ausgegeben. Sie können keine Lademodule mit Lademodus STATIC austauschen (*load_mode*='S'). Lademodule mit Lademodus STARTUP (*load_mode*='U'), die TCB-Entries enthalten, dürfen ebenfalls nicht ausgetauscht werden. In UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen müssen Sie die Version angeben, wenn das Shared Object/DLL mit Lademodus ONCALL generiert ist (*load_mode*='O'). Bei Shared Objects/DLLs mit Lademodus STARTUP (*load_mode*='U') ist die Angabe der Version optional, wenn Sie das Versionskonzept nicht nutzen.

Wirkungsdauer / Transaktionssicherung: Typ GID ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern").

Die Durchführung des Austausches ist abhängig vom Lademodus des Lademoduls/Shared Objects/DLL (Feld *load_mode* in *kc_load_module_str*, siehe Abschnitt "kc_load_module_str - Lademodule (BS2000-Systeme) bzw. Shared Objects/DLLs (Unix-, Linux- und Windows-Systeme)"):

- *load_mode*='U' (STARTUP)
  Der Austausch wird für jeden Prozess vor der Bearbeitung des nächsten Auftrags ausgeführt, ohne dass das laufende Anwendungsprogramm beendet wird. Mehrere Prozesse der Anwendung können gleichzeitig austauschen. Bis der Programmaustausch von allen Prozessen der Anwendung abgeschlossen wurde, dürfen Sie keinen weiteren Austausch veranlassen.
- *load_mode*='O' (ONCALL)
  Der Austausch wird für jeden Prozess erst dann durchgeführt, wenn das nächste Mal in diesem Prozess ein Teilprogramm aus diesem Lademodul bzw. Shared Object/DLL aufgerufen wird. Der Austausch kann von mehreren Prozessen gleichzeitig durchgeführt werden.
- *load_mode*='P', 'T', 'C' (POOL, POOL/STARTUP, POOL/ONCALL, nur auf BS2000-Systemen)

  Der KC_MODIFY_OBJECT-Aufruf bewirkt **nicht** den Austausch des Lademoduls, es wird nur die neue Version des Lademoduls vorgemerkt.

  Den Austausch des Lademoduls müssen Sie explizit anfordern, indem Sie KC_CHANGE_APPLICATION aufrufen oder die Anwendung neu starten. Sie können durch mehrere KC_MODIFY_OBJECT-Aufrufe mehrere Lademodule vormerken, die dann beim folgenden KC_CHANGE_APPLICATION-Aufruf ausgetauscht werden. Folgt im selben Anwendungslauf kein KC_CHANGE_APPLICATION-Aufruf, dann werden die vorgemerkten Versionen beim folgenden Anwendungsstart ausgetauscht.

  Wenn Sie zwischen dem KC_MODIFY_OBJECT-Aufruf und dem KC_CHANGE_APPLICATION-Aufruf einen KC_GET_OBJECT-Aufruf absetzen, dann wird die vorgemerkte Version bereits als aktuelle Version ausgegeben, auch wenn sie noch nicht geladen ist. Der KC_MODIFY_OBJECT-Aufruf bewirkt, dass die neue Version des Lademoduls in den UTM-Tabellen als aktuelle Version und die derzeit geladene Version als Vorgängerversion eingetragen wird. Dem Feld *change_necessary* können Sie entnehmen, ob noch ein Programmaustausch mit KC_CHANGE_APPLICATION nötig ist, um die angegebene Version zu laden.

  KDCPROG ("KDCPROG - Lademodule/Shared Objects/DLLs austauschen")

11.2.9.5 obj_type=KC_LPAP

Die Aktionen beziehen sich auf einen LPAP-Partner, d.h. auf die logischen Eigenschaften einer LU6.1-Partner-Anwendung oder auf die Verbindung zu dieser Partner-Anwendung.

Im Identifikationsbereich müssen Sie den Namen des LPAP-Partners angeben (Feld *kc_name8* der Union *kc_id_area*). Das ist der Name, der bei der KDCDEF-Generierung in der LPAP-Anweisung für die Partner-Anwendung definiert wurde.

Im Datenbereich müssen Sie die Datenstruktur *kc_lpap_str* mit den neuen Eigenschaftswerten übergeben.

Mögliche Modifikationen

- einen LPAP-Partner sperren oder einen gesperrten LPAP-Partner freigeben.

  Über einen gesperrten LPAP-Partner ist keine Verbindung zur Partner-Anwendung mehr möglich.

  In der Datenstruktur *kc_lpap_str* geben Sie Folgendes an:

Feldname	Bedeutung
state='N'	wenn der LPAP-Partner gesperrt werden soll. Zum Zeitpunkt des Sperrens darf keine Verbindung zur Partner-Anwendung existieren. Existierende Verbindungen müssen Sie vor dem Sperren mit *connect_mode*='N' oder *quiet_connect*='Y' abbauen. Der Verbindungsabbau und das Sperren des LPAP-Partners sind nicht in einem Aufruf möglich, da sich der Verbindungsabbau über einen längeren Zeitraum erstrecken kann.
state='Y'	Der LPAP-Partner soll wieder freigegeben werden, d.h. eine existierende Sperre soll aufgehoben werden.

  Wirkungsdauer / Transaktionssicherung: Typ GPD ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

- den automatischen Verbindungsaufbau ein- oder ausschalten.

  Automatischer Verbindungsaufbau heißt, dass UTM beim Start der Anwendung versucht, die Verbindung zur Partner-Anwendung aufzubauen.

  Ist in beiden Anwendungen (lokale Anwendung und Partner-Anwendung) der automatische Verbindungsaufbau definiert, dann wird die Verbindung zwischen beiden automatisch aufgebaut, sobald beide Anwendungen verfügbar sind.

  In der Datenstruktur *kc_lpap_str* geben Sie Folgendes an:

Feldname	Bedeutung
auto_connect='Y'	Ab dem nächsten Anwendungsstart soll UTM versuchen, die Verbindung zur Partner-Anwendung beim Start automatisch aufzubauen.
auto_connect='N'	Ab dem nächsten Start der Anwendung soll die Verbindung zur Partner-Anwendung nicht mehr automatisch aufgebaut werden.

  Wirkungsdauer / Transaktionssicherung: Typ GPD ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

- das Zeitintervall ändern, in dem UTM den Leerlauf-Zustand einer Session zur Partner-Anwendung überwacht. D. h. ist die Session durch keinen Auftrag belegt, wartet UTM dieses Zeitintervall ab, bevor UTM die Verbindung abbaut.

  In der Datenstruktur *kc_lpap_str* geben Sie Folgendes an:

Feldname	Bedeutung
idletime_sec[5]	In *idletime_sec* geben Sie die Zeit in Sekunden an, die UTM den Idle-Zustand einer Session zur Partner-Anwendung überwachen soll. *idletime_sec*='0' bewirkt, dass der Idle-Zustand nicht überwacht wird. Maximalwert: '32767' Minimalwert: '60 Bei Werten ungleich 0 und kleiner als 60 wird der Wert 60 verwendet.

  Wirkungsdauer / Transaktionssicherung: Typ GPD ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

  Die Änderung des Timers wird erst wirksam, wenn die Session den Idle-Zustand das nächste Mal erreicht, aber nicht vor dem Ende des Teilprogrammlaufs (PEND), in dem der Aufruf bearbeitet wird.

- die Verbindung zu der Partner-Anwendung auf- oder abbauen.

  Der Verbindungsabbau kann auf zwei Arten erfolgen:

  - Die Verbindung kann direkt abgebaut werden, d.h. UTM baut die Verbindung ab, unabhängig davon, ob zur Zeit Aufträge auf der Verbindung bearbeitet werden oder nicht (*connect_mode*).
  - Sie können die Verbindung auf QUIET setzen (*quiet_connect*). QUIET bedeutet, dass UTM die Verbindung zur Partner-Anwendung abbaut, sobald die für den LPAP-Partner generierten Sessions nicht mehr durch Aufträge (Dialog- oder Asynchron-Aufträge) belegt sind.

    Es werden für den LPAP-Partner jedoch keine neuen Dialog-Aufträge mehr angenommen. Neue Asynchron-Aufträge werden entgegengenommen, aber nicht mehr gesendet; sie verbleiben in der Ausgabewarteschlange.

Feldname	Bedeutung
connect_mode='Y'	UTM soll die Verbindung zur Partner-Anwendung aufbauen. Ist der LPAP-Partner gesperrt, dann muss er vor dem Verbindungsaufbau in einer eigenen Transaktion freigegeben werden (*state*='Y').
connect_mode='N'	Die Verbindung zur Partner-Anwendung soll sofort abgebaut werden. Beim Verbindungsabbau mit *connect_mode*='N' ist es möglich, dass Vorgänge bzw. Conversations abnormal beendet werden. Sie sollten Verbindungen besser mit *quiet_connect*='Y' abbauen.

  Wirkungsdauer / Transaktionssicherung: Typ A ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

Feldname	Bedeutung
quiet_connect='Y'	Für die Verbindung zur Partner-Anwendung wird die Eigenschaft QUIET gesetzt. Die Eigenschaft QUIET kann mit *connect_mode*='Y' zurückgenommen werden.

Wirkungsdauer / Transaktionssicherung: Typ IR ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

Innerhalb eines Aufrufs können die Felder *connect_mode* und *quiet_connect* nicht gleichzeitig gesetzt werden. Außerdem kann *connect_mode*='Y' nicht gleichzeitig mit *state* ='N' gesetzt werden. Wird für eine Verbindung, die auf QUIET gesetzt ist, ein KC_MODIFY_OBJECT-Aufruf mit *connect_mode*='N' abgesetzt, dann wird die Verbindung sofort abgebaut. *connect_mode*='N' „überschreibt“ *quiet_connect*='Y'.

- den BCAM-Trace für die Verbindung zur Partner-Anwendung ein- bzw. ausschalten.

  Voraussetzung für das LPAP-spezifische Einschalten ist:

  Der BCAM-Trace ist nicht allgemein eingeschaltet, d.h. der Trace ist entweder ganz ausgeschaltet oder nur für ausgewählte LTERM-/LPAP-Partner oder USER explizit eingeschaltet.

  Voraussetzung für das LPAP-spezifische Ausschalten ist:

  Der BCAM-Trace kann nur dann LPAP-spezifisch ausgeschaltet werden, wenn der BCAM-Trace nicht allgemein eingeschaltet ist.

  Informationen zum allgemeinen Ein- und Ausschalten des BCAM-Trace finden Sie bei der Beschreibung der Datenstruktur *kc_diag_and_account_par_str* im Abschnitt "kc_diag_and_account_par_str - Diagnose- und Accounting-Parameter".

Feldname	Bedeutung
bcam_trace='Y'	Der BCAM-Trace wird explizit für diesen LPAP-Partner eingeschaltet. Es werden die Ereignisse auf allen Transportverbindungen zu der Partner-Anwendung aufgezeichnet, der dieser LPAP-Partner zugeordnet ist. Beim Einschalten der Tracefunktion erzeugt jeder Prozess der Anwendung seine eigene Trace-Datei.
bcam_trace='N'	Der BCAM-Trace wird explizit für diesen LPAP-Partner ausgeschaltet. Die Trace-Dateien werden erst geschlossen, wenn der Trace allgemein ausgeschaltet wird (Objekttyp KC_DIAG_AND_ACCOUNT_PAR; "obj_type=KC_DIAG_AND_ACCOUNT_PAR").

  Wirkungsdauer / Transaktionssicherung: Typ IR ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

- Speichern von Asynchron-Nachrichten in der Dead Letter Queue (TAC-Queue KDCDLETQ) festlegen. In der Datenstruktur *kc_tac_str* geben Sie Folgendes an:

Feldname	Bedeutung
dead_letter_q='Y'	Nachrichten an diesen Asynchron-TAC oder diese TAC-Queue, die nicht verarbeitet werden konnten, werden in der Dead Letter Queue gesichert, sofern sie nicht erneut zugestellt werden (Redelivery) und (bei Message-Komplexen) kein negativer Quittungs-Auftrag definiert wurde. *dead_letter_q*='Y' ist nicht erlaubt für KDCDLETQ, KDCMSGTC, alle Dialog-TACs und Asynchron-TACs mit CALL=NEXT.
dead_letter_q='N'	Nachrichten an diesen Asynchron-TAC oder diese TAC-Queue, die nicht verarbeitet werden konnten, werden nicht in der Dead Letter Queue gespeichert, sondern gelöscht. Dieser Wert muss für alle Dialog-TACS und für Asynchron-TACS mit CALL=NEXT, sowie für KDCMSGTC und KDCDLETQ angegeben werden.

- Die Sicherung von Asynchron-Nachrichten für diesen LPAP-Partner in die Dead Letter Queue ein- oder ausschalten. Damit kann verhindert werden, dass Nachrichten für diesen LPAP-Partner bei permanenten Fehlern verloren gehen.

  In der Datenstruktur *kc_lpap_str* geben Sie Folgendes an:

Feldname	Bedeutung
dead_letter_q='Y'	Asynchron-Nachrichten an diesen LPAP-Partner, die wegen eines permanenten Fehlers nicht gesendet werden konnten, werden in die Dead Letter Queue gesichert, sofern (bei Message- Komplexen) kein negativer Quittungsauftrag definiert wurde.
dead_letter_q='N'	Asynchron-Nachrichten an diesen LPAP-Partner, die wegen eines permanenten Fehlers nicht gesendet werden konnten, werden nicht in die Dead Letter Queue gesichert, sondern gelöscht.

  Wirkungsdauer / Transaktionssicherung: Typ GPD ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

> **i** Wenn es sich bei dem LPAP um das Master-LPAP eines LPAP-Bündels handelt, können Sie nur das Feld *state* modifizieren.

KDCLPAP ("KDCLPAP - Verbindungen zu (OSI-)LPAP-Partnern administrieren") / KDCDIAG ("KDCDIAG - Diagnosehilfen ein- und ausschalten") für den BCAM-Trace

11.2.9.6 obj_type=KC_LSES

Die Änderung bezieht sich auf eine Session für die verteilte Verarbeitung über das LU6.1-Protokoll.

Im Identifikationsbereich müssen Sie den Namen der Session (LSES-Name aus der KDCDEF-Generierung) an UTM übergeben (*kc_name8* in der Union *kc_id_area*).

Im Datenbereich müssen Sie die Datenstruktur *kc_lses_str* mit den neuen Eigenschaftswerten übergeben.

Mögliche Modifikationen

- für die Session eine Transportverbindung zur Partner-Anwendung aufbauen.

Feldname	Bedeutung
connect_mode='Y' con, pronam, bcamappl	Es soll eine Transportverbindung für die Session aufgebaut werden. Soll für die Session eine bestimmte Transportverbindung aufgebaut werden, dann müssen Sie diese Transportverbindung in *con*, *pronam*, *bcamappl* eindeutig spezifizieren. Dazu müssen Sie folgende Angaben machen: ○ in *con* den beim Erzeugen oder bei der Generierung des CON-Objekts festgelegten Namen der Verbindung. ○ in *pronam* den Namen des Rechners, auf dem die Partner-Anwendung abläuft. ○ in *bcamappl* den Namen der lokalen UTM-Anwendung (BCAMAPPL-Name), über den die Verbindung zur Partner-Anwendung aufgebaut wird. Geben Sie *con*, *pronam*, *bcamappl* nicht an, dann baut UTM irgendeine der Transportverbindungen auf, die dynamisch oder mit der KDCDEF-Steueranweisung CON für die Partner-Anwendung generiert wurden. Eine Verbindung kann nicht aufgebaut werden, wenn der zugehörige LPAP-Partner gesperrt ist (siehe KC_LPAP *state* ='N' im Abschnitt "obj_type=KC_LPAP"). Ist der LPAP-Partner gesperrt, dann muss er vor dem Verbindungsaufbau mit einem eigenen KC_MODIFY_OBJECT-Aufruf freigegeben werden (KC_LPAP mit *state*='Y').

  Wirkungsdauer / Transaktionssicherung: Typ A ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

- die Transportverbindung, die für die Session besteht, abbauen.

  Sie können UTM veranlassen, die Verbindung sofort abzubauen, oder der Verbindung die Eigenschaft QUIET zuordnen. QUIET bedeutet, dass UTM die Verbindung zur Partner-Anwendung abbaut, sobald die Session nicht mehr durch Aufträge (Dialog- oder Asynchron-Aufträge) belegt ist. Es werden keine neuen Dialog-Aufträge mehr angenommen. Neue Asynchron-Aufträge werden entgegengenommen, aber nicht mehr gesendet; sie verbleiben in der Ausgabewarteschlange.

Feldname	Bedeutung
connect_mode='N'	Die Verbindung zur Partner-Anwendung, die für die Session besteht, soll sofort abgebaut werden. Ein Verbindungsabbau mit *connect_mode*='N' wirkt sofort, deshalb ist es möglich, dass Vorgänge bzw. Conversations abnormal beendet werden. Sie sollten Verbindungen besser mit *quiet_connect*='Y' abbauen.

Wirkungsdauer / Transaktionssicherung: Typ A ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

Feldname	Bedeutung
quiet_connect='Y'	Die Eigenschaft QUIET für die Verbindung zur Partner-Anwendung setzen. Die Eigenschaft QUIET wird zurückgenommen mit *connect_mode*='Y'.

Wirkungsdauer / Transaktionssicherung: Typ IR ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

Zusammen mit *connect_mode*='N' darf kein anderes Feld der Datenstruktur belegt werden. Insbesondere dürfen *connect_mode* und *quiet_connect* nicht gleichzeitig gesetzt werden.

Wird für eine Verbindung *connect_mode*='N' gesetzt, die zuvor auf QUIET gesetzt wurde, dann wird diese Verbindung sofort abgebaut. Die Eigenschaft QUIET wird durch *connect_mode*='N' überschrieben.

KDCLSES ("KDCLSES - Verbindungen für LU6.1-Sessions auf-/abbauen")

11.2.9.7 obj_type=KC_LTAC

Die Änderung bezieht sich auf einen LTAC, d.h. auf einen Transaktionscode der lokalen Anwendung für einen Service in einer Partner-Anwendung.

Im Identifikationsbereich müssen Sie den Namen des LTACs an UTM übergeben (*kc_name8* in der Union *kc_id_area*).

Im Datenbereich müssen Sie die Datenstruktur *kc_ltac_str* mit den neuen Eigenschaftswerten übergeben.

Mögliche Modifikationen

- Sie können die maximale Wartezeit ändern, die UTM beim Anfordern des fernen Service auf das Belegen einer Session wartet. Dazu geben Sie in *kc_ltac_str* Folgendes an:

Feldname	Bedeutung
accesswait_sec[5]	In *accesswait_sec* geben Sie die Zeit in Sekunden an, die UTM nach dem Aufruf des LTACs maximal auf das Belegen einer Session bzw. auf den Aufbau einer Association warten soll. Bei der Angabe der Zeit müssen Sie berücksichtigen, dass die Transportverbindung zur Partner-Anwendung eventuell noch aufgebaut werden muss. *accesswait_sec* != 0 bedeutet bei Asynchron-LTACs, dass der Auftrag immer in die lokale Warteschlange (Message Queue) für die Partner-Anwendung eingetragen wird. Die Wartezeit *accesswait_sec*=0 bedeutet: Bei Dialog-LTACs wird der lokale Vorgang, der den fernen Service aufruft, sofort mit entsprechendem Returncode fortgesetzt, wenn keine Session bzw. Association zur Partner-Anwendung frei ist oder die lokale Anwendung „Contention Loser“ ist (siehe *kc_lpap_str* im Abschnitt "kc_lpap_str - Eigenschaften von LU6.1-Partner-Anwendungen" f; Feld *contwin*). Bei Asynchron-LTACs wird der Asynchron-Auftrag bereits beim FPUT-Aufruf mit einem Returncode abgewiesen, falls zur Partner-Anwendung keine Verbindung besteht. Besteht eine Verbindung zur Partner-Anwendung, dann wird die Nachricht in die Message Queue eingetragen. Minimalwert: '0', Maximalwert: '32767'

  Wirkungsdauer / Transaktionssicherung: Typ GPR ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

- Sie können die Wartezeit ändern, die UTM maximal auf die Anwort des fernen Services wartet. Dazu geben Sie in *kc_ltac_str* Folgendes an:

Feldname	Bedeutung
replywait_sec[5]	In *replywait_sec* geben Sie die Zeit in Sekunden an, die UTM maximal auf die Antwort vom fernen Service warten soll. Durch Begrenzung der Wartezeit kann gewährleistet werden, dass die Wartezeit für Benutzer am Terminal nicht beliebig lang werden kann. *replywait_sec*='0' bedeutet: Warten ohne Zeitbegrenzung. Minimalwert: '0', Maximalwert: '32767'

Wirkungsdauer / Transaktionssicherung: Typ GPR ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

- Sie können den LTAC sperren oder wieder freigeben. Die Sperre eines LTACs bedeutet, dass keine Aufträge für den fernen Service, dem der LTAC zugeordnet ist, aus der lokalen Anwendung mehr angenommen werden. Dazu geben Sie in *kc_ltac_str* Folgendes an:

Feldname	Bedeutung
state='N'	Der LTAC soll gesperrt werden, UTM soll keine Aufträge für den zugehörigen fernen Service mehr annehmen.
state='Y'	Der (gesperrte) LTAC soll freigegeben werden, d.h. die Sperre soll aufgehoben werden.

Wirkungsdauer / Transaktionssicherung: Typ GPR ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

KDCLTAC ("KDCLTAC - Eigenschaften von LTACs ändern")

11.2.9.8 obj_type=KC_LTERM

Die Änderung bezieht sich auf einen LTERM-Partner.

Im Identifikationsbereich müssen Sie den Namen des LTERM-Partners an UTM übergeben (*kc_name8* in der Union *kc_id_area*).

Im Datenbereich müssen Sie die Datenstruktur *kc_lterm_str* mit den neuen Eigenschaftswerten übergeben.

Mögliche Modifikationen

- den LTERM-Partner sperren bzw. gesperrten LTERM-Partner wieder freigeben. LTERM-Partner eines LTERM-Pools können mit *obj_type*=KC_LTERM nicht gesperrt bzw. freigegeben werden (siehe dazu *obj_type* =KC_TPOOL im Abschnitt "obj_type=KC_TPOOL").

  Zum Sperren bzw. Freigeben eines LTERM-Partners geben Sie in *kc_lterm_str* Folgendes an:

Feldname	Bedeutung
state='N'	Sperren des LTERM-Partners. Die Sperre eines Dialog-Partners (*usage_type*='D') wirkt wie folgt: ○ Ein Verbindungswunsch des Client wird ausgeführt. Die Verbindung wird aufgebaut und es wird die Meldung K027 ausgegeben. Es werden außer KDCOFF keine Aufträge des Client /Benutzers ausgeführt. ○ Eine bestehende Verbindung bleibt erhalten. Jede Eingabe mit Ausnahme von KDCOFF wird mit der Meldung K027 quittiert. Die Sperre wirkt erst, wenn auf dieser Verbindung ein Sicherungspunkt (Transaktionsende) erreicht ist. KDCOFF BUT wirkt bei gesperrtem LTERM-Partner wie KDCOFF. Wird der LTERM-Partner eines Druckers gesperrt, bleiben die Druckaufträge in der Message Queue erhalten. Druckaufträge, die nach dem Sperren erzeugt werden, werden nicht abgewiesen. Sie werden in die Message Queue eingetragen.
state='Y'	Freigeben des LTERM-Partners, d.h. Aufheben einer Sperre.

  Wirkungsdauer / Transaktionssicherung: Typ GPD ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

- die Verbindung zum Client oder Drucker, der diesem LTERM-Partner zugeordnet ist, auf- bzw. abbauen.

Feldname	Bedeutung
connect_mode='Y'	Die Verbindung zu dem Client/Drucker soll aufgebaut werden. *connect_mode*='Y' ist nicht erlaubt, wenn der LTERM-Partner, den Sie im Identifikationsbereich angegeben haben, zu einem LTERM-Pool gehört oder einem UPIC-Client zugeordnet ist.
connect_mode='N'	Die Verbindung zum Client/Drucker soll sofort abgebaut werden. Ein mit *connect_mode*='N' initiierter Verbindungsabbau wirkt sofort, deshalb ist es möglich, dass Vorgänge abnormal beendet werden (PEND ER). Mit *connect_mode*='N' können Sie auch die Verbindung zu einem Client abbauen, der über einen LTERM-Pool mit der Anwendung verbunden ist. Dazu geben Sie im Identifikationsbereich den Namen eines LTERM-Partners an, der zu einem LTERM-Pool gehört.

  Wirkungsdauer / Transaktionssicherung: Typ A ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

- Nur auf BS2000-Systemen:
  dem LTERM-Partner ein neues Startformat zuordnen oder das Startformat des LTERM-Partners löschen.

  Ein Startformat können Sie jedem LTERM-Partner zuordnen, der zum Anschluss von Terminals konfiguriert wurde. Zum Ändern des Startformats müssen Sie immer Formatname und Formatattribut des neuen Startformats angeben.

  Voraussetzung für das Zuweisen eines Startformats ist, dass ein Formatierungssystem generiert ist (KDCDEF-Anweisung FORMSYS). Ist das Startformat ein #Format, dann muss zusätzlich ein Anmelde-Vorgang generiert sein.

Feldname	Bedeutung	
format_attr	Formatkennzeichen des neuen Startformats:	
	'A'	für das Formatattribut ATTR. Der Formatname an der Programmschnittstelle KDCS ist +*format_name*.
	'N'	für das Formatattribut NOATTR. Der Formatname an der Programmschnittstelle KDCS ist **format_name*.
	'E'	für das Formatattribut EXTEND. Der Formatname an der Programmschnittstelle KDCS ist #*format_name*.
	Die Bedeutung der Formatattribute ist im im Abschnitt "kc_lterm_str - LTERM-Partner" (format_attr, format_name (nur auf BS2000-Systemen)) beschrieben.	
format_name[7]	Name des Startformats. Der Name kann bis zu 7 Zeichen lang sein und darf nur alphanumerische Zeichen enthalten.	

  Zum Löschen des Startformats geben Sie in *format_attr* und *format_name* Leerzeichen an.

  Wirkungsdauer / Transaktionssicherung: Typ GPD ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

- den BCAM-Trace für die Verbindungen dieses LTERM-Partners ein-/ausschalten.

  Die BCAM-Tracefunktion verfolgt alle Verbindungs-bezogenen Aktivitäten.

Voraussetzung für das LTERM-spezifische Einschalten ist:

Der BCAM-Trace ist nicht allgemein für alle LTERM- und LPAP-Partner eingeschaltet, d.h. der Trace ist entweder ganz ausgeschaltet oder nur für ausgewählte LTERM-/LPAP-Partner oder USER explizit eingeschaltet.

Voraussetzung für das LTERM-spezifische Ausschalten ist:

Der BCAM-Trace kann nur dann LTERM-spezifisch ausgeschaltet werden, wenn der BCAM-Trace nicht allgemein eingeschaltet ist.

Informationen zum allgemeinen Ein- und Ausschalten des BCAM-Trace finden Sie bei der Beschreibung der Datenstruktur *kc_diag_and_account_par_str* im Abschnitt "kc_diag_and_account_par_str - Diagnose- und Accounting-Parameter".

Feldname	Bedeutung
bcam_trace='Y'	Der BCAM-Trace wird explizit für diesen LTERM-Partner eingeschaltet. Es werden alle Ereignisse auf der Verbindung zu dem Client/Drucker protokolliert, der diesem LTERM-Partner zugeordnet ist. Beim Einschalten der Tracefunktion erzeugt jeder Prozess der Anwendung seine eigene Trace-Datei.
bcam_trace='N'	Der BCAM-Trace wird explizit für diesen LTERM-Partner ausgeschaltet. Die Trace-Dateien werden erst geschlossen, wenn der Trace allgemein ausgeschaltet wird (Objekttyp KC_DIAG_AND_ACCOUNT_PAR; "obj_type = KC_DIAG_AND_ACCOUNT_PAR").

Wirkungsdauer / Transaktionssicherung: Typ IR ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

- die Master-LTERMs zweier LTERM-Bündel austauschen oder ein Gruppen-LTERM in eine andere LTERM-Gruppe aufnehmen.

  Diese Funktion ist nur in einer stand-alone UTM-Anwendung erlaubt.

  Handelt es sich bei dem LTERM um das Master-LTERM eines LTERM-Bündels, können Sie alle Slave-LTERMs und die zugehörigen PTERMs mit einem anderen Master-LTERM tauschen. In diesem Fall muss im Parameter *master* ebenfalls ein Master-LTERM eines LTERM-Bündels angegeben werden.

  Handelt es sich bei dem LTERM um ein Gruppen-LTERM einer LTERM-Gruppe, können Sie es einer anderen LTERM-Gruppe zuordnen. Das Primary-LTERM, das Sie im Parameter *master* angeben, muss entweder ein normales LTERM, ein Primary-LTERM einer LTERM-Gruppe oder das Master-LTERM eines LTERM-Bündels sein. Ein normales LTERM muss dabei folgende Bedingungen erfüllen:

  - Dem LTERM muss ein PTERM mit PTYPE APPLI oder SOCKET zugeordnet sein.
  - Das LTERM darf kein Slave-LTERM eines LTERM-Bündels sein.
  - Das LTERM muss mit USAGE=D generiert sein.

  In der Datenstruktur *kc_lterm_str* geben Sie Folgendes an:

Feldname	Bedeutung
master[8]	Name eines Master-LTERMs in einem LTERM-Bündel, Name eines Primary-LTERMs in einer LTERM-Gruppe oder Name eines normalen LTERMs. Der Name kann bis zu 8 Zeichen lang sein und darf nur alphanumerische Zeichen enthalten.

  Wirkungsdauer / Transaktionssicherung: Typ PD ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

  Einige der Modifikationen sind auch mit KDCLTERM ("KDCLTERM - Eigenschaften von LTERM-Partnern ändern") bzw. KDCDIAG ("KDCDIAG - Diagnosehilfen ein- und ausschalten") durchführbar.

11.2.9.9 obj_type=KC_MUX (BS2000-Systeme)

Die Aktion bezieht sich auf einen Multiplexanschluss.

Im Identifikationsbereich müssen Sie den Multiplexanschluss eindeutig identifizieren. Dazu übergeben Sie in der Datenstruktur *kc_triple_str* der Union *kc_id_area* den Namen des Multiplexanschlusses, den Namen des Rechners, an dem sich der zugehörige Nachrichtenverteiler befindet, und den Namen der UTM-Anwendung, über den die Multiplexverbindung aufgebaut wird.

Im Datenbereich müssen Sie die Datenstruktur *kc_mux_str* mit den neuen Eigenschaftswerten übergeben.

Mögliche Modifikationen

- einen Multiplexanschluss sperren bzw. einen gesperrten Multiplexanschluss wieder freigeben.

  Über einen gesperrten Multiplexanschluss kann keine Verbindung zwischen Nachrichtenverteiler und UTM-Anwendung aufgebaut werden. In der Datenstruktur *kc_mux_str* geben Sie Folgendes an:

Feldname	Bedeutung
state='N'	Sperren eines Multiplexanschlusses Zum Zeitpunkt des Sperrens darf keine Verbindung zu dem Multiplexanschluss existieren. Existierende Verbindungen müssen Sie vor dem Sperren mit *connect_mode*='N' abbauen. Der Verbindungsabbau und das Sperren eines Multiplexanschlusses sind nicht in einem KC_MODIFY_OBJECT-Aufruf möglich, weil der Verbindungsabbau einige Zeit in Anspruch nimmt.
state='Y'	Zum Freigeben eines Multiplexanschlusses, d.h. zum Aufheben einer Sperre.

  Wirkungsdauer / Transaktionssicherung: Typ GPD ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

- die Anzahl der Clients, die sich maximal gleichzeitig über diesen Multiplexanschluss anschließen können, herauf- oder herabsetzen.

Feldname	Bedeutung
maxses[5]	In *maxses* geben Sie an, wie viele Sessions maximal gleichzeitig zwischen Nachrichtenverteiler und Anwendung bestehen dürfen. Minimalwert:'1', Maximalwert:'65000' (theoretischer Wert)

  Wirkungsdauer / Transaktionssicherung: Typ GPR ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

- den automatischen Verbindungsaufbau zum Multiplexanschluss ein- oder ausschalten.

  Beim automatischen Verbindungsaufbau versucht UTM beim Start der Anwendung die Verbindung zum Multiplexanschluss automatisch aufzubauen.

  In der Datenstruktur *kc_mux_str* geben Sie Folgendes an:

Feldname	Bedeutung
auto_connect='Y'	UTM soll ab dem nächsten Start der Anwendung die Verbindung zum Multiplexanschluss automatisch aufbauen.
auto_connect='N'	Ab dem nächsten Start der Anwendung soll UTM die Verbindung zum Multiplexanschluss nicht mehr automatisch aufbauen. Sie muss dann vom Administrator explizit aufgebaut werden.

  Wirkungsdauer / Transaktionssicherung: Typ GPD ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

- die Verbindung zu dem Nachrichtenverteiler des Multiplexanschlusses auf- bzw. abbauen.

  In der Datenstruktur *kc_mux_str* geben Sie Folgendes an:

Feldname	Bedeutung
connect_mode='Y'	UTM soll die Verbindung zum Nachrichtenverteiler aufbauen. Soll eine Verbindung für einen gesperrten Multiplexanschluss aufgebaut werden, dann muss der Multiplexanschluss vor dem Verbindungsaufbau mit einem eigenen KC_MODIFY_OBJECT-Aufruf freigegeben werden (*state*='Y'). *connect_mode*='Y' kann nicht gleichzeitig mit *state* ='N' (Multiplexanschluss sperren) gesetzt werden.
connect_mode='N'	Die Verbindung zum Nachrichtenverteiler soll sofort abgebaut werden. Ein mit *connect_mode*='N' initiierter Verbindungsabbau wirkt sofort, deshalb ist es möglich, dass Sessions abnormal beendet werden.

  Wirkungsdauer / Transaktionssicherung: Typ A ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

- BCAM-Trace für diesen Multiplexanschluss explizit ein- oder ausschalten. In der Datenstruktur *kc_mux_str* geben Sie Folgendes an:

Feldname	Bedeutung
bcam_trace='Y'	Der BCAM-Trace wird explizit für diesen Multiplexanschluss eingeschaltet. Es werden alle Ereignisse auf der Verbindung zum Nachrichtenverteiler aufgezeichnet, der diesem Multiplexanschluss zugeordnet ist. Beim Einschalten der Tracefunktion erzeugt jeder Prozess der Anwendung seine eigene Trace-Datei.
bcam_trace='N'	Der BCAM-Trace wird explizit für diesen Multiplexanschluss ausgeschaltet. Die Trace-Dateien werden erst geschlossen, wenn der Trace allgemein ausgeschaltet wird (Objekttyp KC_DIAG_AND_ACCOUNT_PAR im Abschnitt "obj_type = KC_DIAG_AND_ACCOUNT_PAR").

Wirkungsdauer / Transaktionssicherung: Typ IR ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

KDCMUX ("KDCMUX - Eigenschaften von Multiplex-Anschlüssen ändern (BS2000- Systeme)") / KDCDIAG ("KDCDIAG - Diagnosehilfen ein- und ausschalten") für den BCAM-Trace.

11.2.9.10 obj_type=KC_OSI_CON

Die Aktion bezieht sich auf eine Verbindung für die verteilte Verarbeitung über OSI TP.
Im Identifikationsbereich müssen Sie den Namen der Verbindung angeben, der bei der KDCDEF-Generierung in OSI-CON definiert wurde (Feld *kc_name8* der Union *kc_id_area*).

Im Datenbereich müssen Sie die Datenstruktur *kc_osi_con_str* mit den neuen Eigenschaftswerten übergeben.

Mögliche Modifikation

Sie können eine Ersatzverbindung (inaktiv gesetzte Verbindung) zu einer OSI TP-Partner-Anwendung aktivieren. In der Datenstruktur *kc_osi_con_str* geben Sie Folgendes an:

Feldname	Bedeutung
active='Y'	UTM soll die Ersatzverbindung aktivieren. Bevor UTM die Ersatzverbindung aktiviert, deaktiviert UTM die zuvor aktive Verbindung zur Partner-Anwendung. Beim Aktivieren der Ersatzverbindung darf deshalb keine Association zu der zugehörigen Partner-Anwendung bestehen.

Wirkungsdauer / Transaktionssicherung: Typ GIR ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

KDCLPAP ("KDCLPAP - Verbindungen zu (OSI-)LPAP-Partnern administrieren") Operand OSI-CON

11.2.9.11 obj_type=KC_OSI_LPAP

Die Aktion bezieht sich auf einen OSI-LPAP-Partner, d.h. auf die logischen Eigenschaften einer OSI TP-Partner-Anwendung oder auf die Verbindung zu dieser Partner-Anwendung.

Im Identifikationsbereich müssen Sie den Namen des zugehörigen OSI-LPAP-Partners angeben (Feld *kc_name8* der Union *kc_id_area*). Der Name wird bei der KDCDEF-Generierung in der OSI-LPAP-Anweisung für die Partner-Anwendung definiert.

Im Datenbereich müssen Sie die Datenstruktur *kc_osi_lpap_str* mit den neuen Eigenschaftswerten übergeben.

Mögliche Modifikationen

> **i** Wenn es sich bei dem OSI-LPAP um das Master-LPAP eines OSI-LPAP-Bündels handelt, können Sie nur das Feld *state* modifizieren.

- einen OSI-LPAP-Partner sperren oder einen gesperrten OSI-LPAP-Partner freigeben. Über einen gesperrten OSI-LPAP-Partner ist keine Verbindung zur Partner-Anwendung mehr möglich.

  In der Datenstruktur *kc_osi_lpap_str* geben Sie Folgendes an:

Feldname	Bedeutung
state='N'	Der OSI-LPAP-Partner soll gesperrt werden. Zum Zeitpunkt des Sperrens darf keine Verbindung zur Partner-Anwendung existieren. Existierende Verbindungen müssen Sie vor dem Sperren mit einem eigenen Aufruf mit *connect_number*='0' oder *quiet_connect*='Y' abbauen. Der Verbindungsabbau und das Sperren des OSI-LPAP-Partners sind nicht innerhalb einer Transaktion möglich.
state='Y'	Der OSI-LPAP-Partner soll wieder freigegeben werden, d.h. existiert eine Sperre, dann soll sie aufgehoben werden.

  Wirkungsdauer / Transaktionssicherung: Typ GPD ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

- die Anzahl der Verbindungen zur Partner-Anwendung, die UTM beim Start der Anwendung automatisch aufbaut, herauf- bzw. herabsetzen.

  Beim automatischen Verbindungsaufbau versucht UTM beim Start der Anwendung die gewünschte Anzahl an Verbindungen zur Partner-Anwendung aufzubauen.

  Ist in beiden Anwendungen (lokale Anwendung und Partner-Anwendung) der automatische Verbindungsaufbau definiert, dann wird die Verbindung zwischen beiden automatisch aufgebaut, sobald beide Anwendungen verfügbar sind.

  In der Datenstruktur *kc_osi_lpap_str* geben Sie Folgendes an:

Feldname	**Bedeutung**
auto_connect_number	In *auto_connect_number* geben Sie die Anzahl der Verbindungen zur Partner-Anwendung an, die UTM beim nächsten Start der Anwendung (und den folgenden Starts) automatisch aufbauen soll. Der OSI-LPAP-Partner, über den sich die Partner-Anwendung anschließt, darf nicht gesperrt sein. Geben Sie *auto_connect_number*='0' an, dann findet beim folgenden Anwendungsstart kein automatischer Verbindungsaufbau statt. Wird eine Anzahl angegeben, die größer ist als die generierte maximale Anzahl paralleler Verbindungen (siehe Feld *associations* in *kc_osi_lpap_str*), dann versucht UTM beim nächsten Start alle generierten parallelen Verbindungen (= Anzahl in *associations*) aufzubauen. Der in *auto_connect_number* angegebene Wert muss jedoch kleiner oder gleich '32767' sein. Minimalwert: '0' Maximalwert: generierte Maximalzahl paralleler Verbindungen (*associations*)

  Wirkungsdauer / Transaktionssicherung: Typ GPD ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

- die Anzahl der parallelen Verbindungen, die aktuell zwischen UTM-Anwendung und Partner-Anwendung existieren sollen, herauf- oder herabsetzen. D.h. Sie können veranlassen, dass zusätzliche Verbindungen aufgebaut werden oder ein Teil der existierenden Verbindungen abgebaut wird. Der Aufbau zusätzlicher Verbindungen ist nur möglich, wenn noch nicht die mit KDCDEF generierte maximale Anzahl paralleler Verbindungen zur Partner-Anwendung aufgebaut ist.

  In der Datenstruktur *kc_osi_lpap_str* geben Sie Folgendes an:

Feldname	Bedeutung
connect_number	In *connect_number* geben Sie die Anzahl der Verbindungen an, die insgesamt zur Partner-Anwendung bestehen sollen. Die Wirkung des Aufrufs ist somit abhängig von der Angabe für *connect_number*. Folgende Fälle müssen unterschieden werden: ○ Geben Sie in *connect_number* eine Anzahl an, die kleiner ist als die Anzahl der derzeit aufgebauten parallelen Verbindungen, dann baut UTM soviele Verbindungen zu der Partner-Anwendung ab, bis nur noch *connect_number* Verbindungen existieren. UTM baut zunächst die Verbindungen ab, die nicht durch Aufträge belegt sind. Existieren danach noch mehr Verbindungen, als in *connect_number* angegeben, dann baut UTM auch Verbindungen ab, die durch Aufträge belegt sind. Die derzeit auf den Verbindungen aktiven Vorgänge bzw. Conversations werden abgebrochen. Geben Sie *connect_number*='0' an, dann baut UTM alle Verbindungen zur Partner-Anwendung ab. ○ Geben Sie in *connect_number* eine Anzahl an, die größer ist als die Anzahl der derzeit aufgebauten parallelen Verbindungen, dann versucht UTM weitere Verbindungen zur Partner-Anwendung aufzubauen, bis insgesamt *connect_number* Verbindungen existieren. UTM baut jedoch maximal soviele parallele Verbindungen zur Partner-Anwendung auf, wie bei der KDCDEF-Generierung für den zur Partner-Anwendung gehörenden OSI-LPAP-Partner festgelegt wurden. Diese Maximalzahl wird bei der Informationsabfrage im Feld *associations* von *kc_osi_lpap_str* zurückgeliefert. D.h. ist *connect_number* > *associations*, dann baut UTM nur die generierte Maximalzahl an Verbindungen auf.
	Sollen Verbindungen zu einem gesperrten OSI-LPAP-Partner aufgebaut werden, müssen Sie diesen zuvor entsperren (siehe Feld *state* im Abschnitt "obj_type=KC_OSI_LPAP"). Die Freigabe des OSI-LPAP-Partners muss in einem eigenen KC_MODIFY_OBJECT-Aufruf erfolgen. *connect_number* und *quiet_connect* dürfen nicht zusammen in einem KC_MODIFY_OBJECT-Aufruf angegeben werden. *connect_number* darf auch nicht zusammen mit *state*='N' angegeben werden.
	Minimalwert: '0' Maximalwert: die von UTM in *associations* zurückgelieferte Anzahl; ein Zahlenwert größer als '32767' wird zurückgewiesen.

  Wirkungsdauer / Transaktionssicherung: Typ A ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

- alle parallelen Verbindungen zur Partner-Anwendung abbauen.

  Sie können UTM veranlassen, alle Verbindungen sofort abzubauen, oder den Verbindungen die Eigenschaft QUIET zuzuordnen. QUIET bedeutet, dass UTM die Verbindungen zur Partner-Anwendung abbaut, sobald diese nicht mehr durch Aufträge (Dialog- oder Asynchron-Aufträge) belegt sind. Es werden keine neuen Dialog-Aufträge mehr angenommen. Neue Asynchron-Aufträge werden entgegengenommen, aber nicht mehr gesendet; sie verbleiben in der Ausgabewarteschlange.

Feldname	**Bedeutung**
connect_number='0'	Geben Sie *connect_number*='0' an, dann baut UTM alle Verbindungen zur Partner-Anwendung sofort ab, auch wenn auf den Verbindungen noch Vorgänge bzw. Conversations aktiv sind. Sie sollten Verbindungen deshalb besser mit *quiet_connect* ='Y' verzögert abbauen.

Wirkungsdauer / Transaktionssicherung: Typ A ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

Feldname	**Bedeutung**
quiet_connect='Y'	Für die Verbindungen zur Partner-Anwendung wird die Eigenschaft QUIET gesetzt. Die Eigenschaft QUIET kann mit *connect_number* > '0' zurückgesetzt werden.

Wirkungsdauer / Transaktionssicherung: Typ IR ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

Innerhalb eines KC_MODIFY_OBJECT-Aufrufs können *connect_number* und *quiet_connect* nicht gleichzeitig gesetzt werden.

- Das Zeitintervall, in dem der Leerlauf-Zustand der Associations von der UTM-Anwendung zur Partner-Anwendung überwacht wird, ändern.

  Wird die Association innerhalb dieses Zeitintervalls nicht durch einen Auftrag belegt, dann baut UTM die Verbindung ab. In der Datenstruktur *kc_osi_lpap_str* geben Sie Folgendes an:

Feldname	Bedeutung
idletime_sec[5]	In *idletime_sec* geben Sie die Zeit in Sekunden an, die UTM den Idle-Zustand einer Association zur Partner-Anwendung überwachen soll. *idletime_sec*='0' bewirkt, dass der Idle-Zustand nicht überwacht wird. Maximalwert: '32767' Minimalwert: '60 Bei Werten ungleich 0 und kleiner als 60 wird der Wert 60 verwendet.

  Wirkungsdauer / Transaktionssicherung: Typ GPD ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

  Die Änderung des Timers wird erst wirksam, wenn die Association den Idle-Zustand das nächste Mal erreicht, aber nicht vor dem Ende des Teilprogrammlaufs (PEND), in dem der Aufruf bearbeitet wird.

- Die Sicherung von Asynchron-Nachrichten für diesen OSI-LPAP-Partner in die Dead Letter Queue ein- oder ausschalten. Damit kann verhindert werden, dass Nachrichten für diesen OSI-LPAP-Partner bei permanenten Fehlern verloren gehen.

  In der Datenstruktur *kc_osi_lpap_str* geben Sie Folgendes an:

Feldname	Bedeutung
dead_letter_q='Y'	Asynchron-Nachrichten an diesen OSI-LPAP-Partner, die wegen eines permanenten Fehlers nicht gesendet werden konnten, werden in die Dead Letter Queue gesichert, sofern (bei Message- Komplexen) kein negativer Quittungsauftrag definiert wurde.
dead_letter_q='N'	Asynchron-Nachrichten an diesen OSI-LPAP-Partner, die wegen eines permanenten Fehlers nicht gesendet werden konnten, werden nicht in die Dead Letter Queue gesichert, sondern gelöscht.

  Wirkungsdauer / Transaktionssicherung: Typ GPD ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

  KDCLPAP ("KDCLPAP - Verbindungen zu (OSI-)LPAP-Partnern administrieren")

11.2.9.12 obj_type=KC_PTERM

Die Aktion bezieht sich auf einen Client oder Drucker der Anwendung.

Im Identifikationsbereich müssen Sie den Client/Drucker eindeutig identifizieren. Dazu übergeben Sie in der Datenstruktur *kc_long_triple_str* der Union *kc_id_area* den Namen des Client/Druckers, den Namen des Rechners, auf dem er sich befindet, und den Namen der UTM-Anwendung, über den die Verbindung aufgebaut wird.

Im Datenbereich müssen Sie die Datenstruktur *kc_pterm_str* mit den neuen Eigenschaftswerten übergeben.

Mögliche Modifikationen

- die Zuordnung Client/Drucker zu LTERM-Partner ändern.

  Damit können Sie die logischen Eigenschaften des Client/Druckers ändern. Insbesondere können Sie damit einen Drucker einem Druckerbündel oder einem Druckersteuerterminal zuordnen. Beim Ändern der Zuordnung dürfen weder Client/Drucker noch der LTERM-Partner, dem der Client/Drucker zugeordnet werden soll, mit der Anwendung verbunden sein.

  Einschränkung:

  Eine neue Zuordnung des LTERM-Partners ist nur für Terminals und Drucker möglich. Bei UPIC-Clients, bei TS-Anwendungen (APPLI/SOCKET), die als Dialog-Partner generiert sind, und bei Clients, die sich über einen LTERM-Pool an die Anwendung anbinden, kann die bei der Konfiguration festgelegte Zuordnung zu einem LTERM-Partner nicht geändert werden.

  Wenn Sie einem Terminal oder einem Drucker einen neuen LTERM-Partner zuweisen, dann darf der LTERM-Partner keinem Client/Drucker eines anderen Protokoll-Typs zugewiesen sein (weder aktuell noch in der Vergangenheit). Dabei sind hier die vier Protokoll-Typen Terminals, TS-Anwendungen, Drucker und RSO-Drucker zu unterscheiden. Es ist z.B. nicht möglich,

  - einem Terminal einen LTERM-Partner zuzuordnen, der einem UPIC-Client oder einer TS-Anwendung zugeordnet ist oder war,
  - auf einem BS2000-System einem RSO-Drucker einen LTERM-Partner zuzuordnen, der einem normalen Drucker zugeordnet ist oder war (und umgekehrt).

Feldname	**Angaben**
lterm[8]	In *lterm* geben Sie den Namen des LTERM-Partners an, der diesem Client/Drucker zugeordnet werden soll. Diese Funktion ist nur in einer stand-alone UTM-Anwendung erlaubt. Der LTERM-Partner muss in der Konfiguration der Anwendung existieren. Es darf kein LTERM-Partner eines LTERM-Pools, kein Master- oder Slave-LTERM eines LTERM-Bündels und kein Gruppen- oder Primary-LTERM einer LTERM-Gruppe sein. Der Name ist maximal 8 Zeichen lang. Bei Clients wird die alte Zuordnung dieses LTERM-Partners implizit aufgelöst. Drucker dürfen Sie nur LTERM-Partnern zuordnen, die für die Ausgabe konfiguriert wurden (*usage_type*='O'). Bei Druckern wird die alte Zuordnung des in *lterm* angegebenen LTERM-Partners nicht aufgelöst, wenn diesem zuvor ein Drucker zugeordnet war. Beide Drucker werden zu einem Druckerbündel zusammengefasst. Zu einem Druckerbündel dürfen beliebig viele Drucker gehören. Ist der LTERM-Partner einem Druckersteuer-LTERM zugeordnet, dann muss der Drucker eine Drucker-Id haben, die im Bereich des Druckersteuer-LTERMs eindeutig ist, sonst wird der Aufruf abgewiesen. *connect_mode* und *lterm* dürfen nicht zusammen in einem Aufruf angegeben werden.

Wirkungsdauer / Transaktionssicherung: Typ PD ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

- den automatischen Aufbau der Verbindung zu dem Client/Drucker ein- oder ausschalten.

  Beim automatischen Verbindungsaufbau versucht UTM beim Start der Anwendung die Verbindung zum Client /Drucker automatisch aufzubauen.

  Ausnahme:

  Zu Clients, die sich über einen LTERM-Pool an die Anwendung anschließen, und zu UPIC-Clients kann kein automatischer Verbindungsaufbau stattfinden. In beiden Fällen geht die Initiative zum Verbindungsaufbau immer vom Client aus, nicht von der UTM-Anwendung.

  In der Datenstruktur *kc_pterm_str* geben Sie Folgendes an:

Feldname	**Bedeutung**
auto_connect='Y'	UTM soll ab dem nächsten Start der Anwendung die Verbindung zum Client/Drucker automatisch aufbauen, sofern der Client/Drucker verfügbar ist. Der Client/Drucker darf nicht gesperrt sein (*state*='N').
auto_connect='N'	Ab dem nächsten Start der Anwendung soll UTM die Verbindung zum Client/Drucker nicht mehr automatisch aufbauen.

  Wirkungsdauer / Transaktionssicherung: Typ GPD ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

- einen Client oder Drucker sperren bzw. eine gesetzte Sperre zurücknehmen.

  Gesperrt werden können nur Clients und Drucker, die explizit mit einer PTERM-Anweisung statisch oder als Objekt vom Typ KC_PTERM dynamisch in die Konfiguration eingetragen wurden. Clients, die sich über einen LTERM-Pool oder einen Multiplexanschluss anschließen, können nicht gesperrt werden.

  Zum Sperren bzw. Freigeben eines Client/Druckers geben Sie in *kc_pterm_str* Folgendes an:

Feldname	**Bedeutung**
state='N'	Den Client/Drucker sperren. Eine Sperre für einen Client wirkt erst beim nächsten Versuch des Client, eine Verbindung zur UTM-Anwendung aufzubauen. Der Verbindungswunsch wird dann von UTM abgelehnt. Eine zum Zeitpunkt des Sperrens bestehende Verbindung bleibt erhalten.
state='Y'	Die Sperre des Client/Drucker soll aufgehoben werden.

  Wirkungsdauer / Transaktionssicherung: Typ GPD ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

- die Verbindung zum Client/Drucker auf- bzw. abbauen.

Feldname	Bedeutung
connect_mode='Y'	Die Verbindung zu dem Client/Drucker soll aufgebaut werden. Ausnahme: *connect_mode*='Y' darf nicht für Clients angegeben werden, die über einen LTERM-Pool mit der Anwendung verbunden sind, und nicht für UPIC-Clients. Der Client/Drucker darf nicht gesperrt sein. Ein gesperrter Client/Drucker muss vor dem Verbindungsaufbau freigegeben werden (*state*='Y'). Freigabe des Client/Druckers und Verbindungsaufbau können nicht in einem Aufruf ausgeführt werden.
connect_mode='N'	Die Verbindung zum Client/Drucker soll sofort abgebaut werden. Ein mit *connect_mode*='N' initiierter Verbindungsabbau wirkt sofort. Ist die Verbindung zu diesem Zeitpunkt durch einen Auftrag belegt, dann wird die Bearbeitung des Auftrags abgebrochen.
connect_mode='R'	*Nur auf BS2000-Systemen:* Darf nur für Clients angegeben werden, die über einen Multiplexanschluss mit einer UTM-Anwendung auf einem BS2000-System verbunden sind. *connect_mode*='R' (Release pending connections) veranlasst UTM, eine Session im Zustand DISCONNECT PENDING mit abgelaufenem Timer freizugeben. Die Session kann nicht freigegeben werden, wenn der Timer noch nicht abgelaufen ist. Zum Zustand DISCONNECT PENDING siehe openUTM-Handbuch „Anwendungen generieren“.

connect_mode und *lterm* dürfen nicht zusammen in einem Aufruf angegeben werden.

Wirkungsdauer / Transaktionssicherung: Typ A ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

- das Zeitintervall ändern, in dem UTM nach dem Ende einer Transaktion oder nach dem Anmelden maximal auf eine Eingabe vom Client wartet. Bei Zeitüberschreitung wird die Verbindung zum Client abgebaut (nur relevant bei Dialog-Partnern).

  In der Datenstruktur *kc_pterm_str* geben Sie Folgendes an:

Feldname	Angaben
idletime[5]	In *idletime* geben Sie die Zeit in Sekunden an, die UTM außerhalb einer Transaktion, d. h. nach dem Ende einer Transaktion oder nach dem Anmelden, maximal auf eine Eingabe vom Client wartet. *idletime*=0 bewirkt, dass unbegrenzt gewartet wird. Maximalwert: '32767' Minimalwert: '60 Bei Werten ungleich 0 und kleiner als 60 wird der Wert 60 verwendet.

  Wirkungsdauer / Transaktionssicherung: Typ GPD ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

  Die Änderung des Timers wird erst beim nächsten Transaktionsende wirksam, aber nicht vor dem Ende des Teilprogrammlaufs (PEND), in dem der Aufruf bearbeitet wird.

  KDCPTERM ("KDCPTERM - Eigenschaften von Clients und Druckern ändern") mit Ausnahme von *idletime*.

11.2.9.13 obj_type=KC_TAC

Die Aktion bezieht sich auf den Transaktionscode eines lokalen Services (*tac_type*='A' oder 'D') oder eine TAC-Queue (*tac_type*='Q').

Im Identifikationsbereich müssen Sie den Namen des Transaktionscodes bzw. der TAC-Queue (Feld *kc_name8* der Union *kc_id_area*) und im Datenbereich die Datenstruktur *kc_tac_str* mit den neuen Eigenschaftswerten übergeben.

Für Transaktionscodes und TAC-Queues können Sie den Status und den Zugriffsschutz ändern. Für Transaktionscodes können Sie auch TAC-spezifische Statistikwerte auf 0 zurücksetzen. Statistikwerte und Statuswerte können jedoch nicht innerhalb eines KC_MODIFY_OBJECT-Aufrufs verändert werden.

Mögliche Modifikationen

- Status eines Transaktionscodes oder einer TAC-Queue ändern.

  Sie können einen Transaktionscode bzw. eine TAC-Queue sperren oder einen gesperrten Transaktionscode bzw. eine gesperrte TAC-Queue wieder zulassen.

  Das Administrationskommando KDCTAC kann nicht gesperrt werden.

  Wenn Sie innerhalb eines Aufrufs den Status eines Transaktionscodes ändern, schließt das ein Zurücksetzen von Statistikwerten aus.

  Zum Sperren bzw. Entsperren geben Sie in *kc_tac_str* Folgendes an:

Feldname	Bedeutung
state='N'	Der Transaktionscode bzw. die TAC-Queue soll gesperrt werden. Die Sperre bedeutet, dass UTM keine weiteren Aufträge für diesen Transaktionscode bzw. für die TAC-Queue annimmt. ○ *tac_type*='A' oder 'D': Der Transaktionscode ist als Vorgangs-TAC (Erster TAC eines Vorgangs) gesperrt. Als Folge-TAC in einem Vorgang (*call_type*='B') ist er nicht gesperrt. Asynchron-Aufträge, die zum Zeitpunkt des Sperrens in der Message Queue des Transaktionscodes stehen, werden noch gestartet. ○ *tac_type*='Q': Die TAC-Queue wird für Schreibzugriffe gesperrt; Lesezugriffe sind möglich. Mit *state*='N' können Sie keine Transaktionscodes sperren, für die *call_type*='N' gesetzt ist.
state='H'	Der Transaktionscode bzw. die TAC-Queue soll vollständig gesperrt werden (Halt). ○ *tac_type*='A' oder 'D': Der Transaktionscode wird als Vorgangs-TAC und auch als Folge-TAC in einem Asynchron- oder Dialog-Vorgang gesperrt. Asynchron-Aufträge, die zum Zeitpunkt des Sperrens in der Message Queue des Transaktionscodes stehen, werden nicht mehr gestartet. Sie bleiben in der Queue, bis der Transaktionscode wieder freigegeben bzw. auf *state*='N' gesetzt wird. ○ *tac_type*='Q': Die TAC-Queue wird für Schreib- und Lesezugriffe gesperrt.

Feldname	Bedeutung
state='K'	Darf nur für Asynchron-Transaktionscodes (*tac_type*='A') angegeben werden, die auch Vorgangs-TACs (*call_type*='B'oder 'F') sind, und für TAC-Queues. Der Transaktionscode bzw. die TAC-Queue wird gesperrt. ○ *tac_type*='A': Aufträge für den Transaktionscode werden zwar angenommen, jedoch nicht bearbeitet. Die Aufträge werden lediglich in die Auftragswarteschlange des Transaktionscodes geschrieben. Die Aufträge werden erst bearbeitet, wenn Sie den Status des Transaktionscodes ändern in 'Y' oder 'N'. ○ *tac_type*='Q': Die TAC-Queue ist für Lesezugriffe gesperrt; Schreiben ist weiterhin möglich. *state*='K' (Keep) können Sie benutzen, um Aufträge zu sammeln, die erst zu einem Zeitpunkt ausgeführt werden sollen, an dem die Anwendung weniger belastet ist (z.B. nachts).
state='Y'	Der Transaktionscode bzw. die TAC-Queue soll wieder zugelassen werden. *state*='Y' setzt sowohl *state*='N', *state*='H' als auch *state*='K' zurück.

Wirkungsdauer / Transaktionssicherung: Typ GID ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

Wird der Transaktionscode KDCMSGTC gesperrt, dann werden alle Meldungen mit Meldungsziel MSGTAC, die sich noch im Pagepool befinden, gelöscht.

KDCTAC ("KDCTAC - Transaktionscodes und TAC-Queues sperren, wieder freigeben")

- Statistikinformationen zu dem Transaktionscode auf 0 zurücksetzen.
  Sie können die Statistikwerte im laufenden Betrieb auf 0 zurücksetzen, indem Sie in *kc_tac_str* eines der folgenden Felder mit 0 versorgen. UTM setzt dann alle Felder auf 0 zurück. Ein Wert != 0 wird zurückgewiesen.

Feldname	**Bedeutung**
used	Anzahl der Teilprogrammläufe mit diesem Transaktionscode.
number_errors	Anzahl der Teilprogrammläufe, die fehlerhaft beendet wurden.
db_counter	Mittlere Anzahl der Datenbankaufrufe aus Teilprogrammen, die über diesen Transaktionscode gestartet wurden.
tac_elap_msec	Mittlere Laufzeit der Teilprogramme, die über diesen Transaktionscode gestartet wurden (elapsed time)
db_elap_msec	Mittlere Zeit, die für die Bearbeitung von Datenbankaufrufen in den Teilprogrammläufen mit diesem TAC benötigt wurde.
taccpu_msec	Durchschnittliche CPU-Zeit in Millisekunden, die zur Bearbeitung dieses Transaktionscodes im Teilprogramm verbraucht wurde. Dies entspricht der von UTM verbrauchten CPU-Zeit plus der vom Datenbanksystem verbrauchten CPU-Zeit.
taccpu_micro_sec	Durchschnittliche CPU-Zeit in Mikrosekunden, die zur Bearbeitung dieses Transaktionscodes im Teilprogramm verbraucht wurde. Dies entspricht der von UTM verbrauchten CPU-Zeit plus der vom Datenbanksystem verbrauchten CPU-Zeit.
nbr_ta_commits	Anzahl der Teilprogrammläufe zu diesem TAC, die eine Transaktion erfolgreich abgeschlossen haben.
number_errors_ex	siehe *number_errors*.

Sie können die Statistikwerte entweder für einen bestimmten Transaktionscode zurücksetzen oder für alle Transaktionscodes der Anwendung. Wollen Sie die Werte für einen Transaktionscode zurücksetzen, dann müssen Sie im Identifikationsbereich den Namen des Transaktionscodes angeben. Im anderen Fall ist der Identifikationsbereich mit binär null zu versorgen.

Wirkungsdauer / Transaktionssicherung: Typ GIR ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

- Sie können den Zugriffsschutz für einen Transaktionscode verändern. War der Transaktionscode bisher durch einen Lockcode geschützt, können Sie den Lockcode entfernen und den Zugriffsschutz über eine Access Liste regeln und umgekehrt. Beachten Sie bitte, dass Lockcode und Access Liste einander ausschließen; es ist immer nur eine Art des Zugriffsschutzes erlaubt.

Feldname	Bedeutung
lock_code[4]	*lock_code* kann eine Zahl zwischen '0' und der Obergrenze sein, die in der MAX-Anweisung, Operand KEYVALUE definiert wurde. Mit '0' entfernen Sie den Zugriffsschutz .
access_list[8]	in *access_list* können Sie ein existierendes Keyset angeben oder das Feld mit Leerzeichen füllen. Mit Leerzeichen entfernen Sie den Zugriffsschutz.

  Ein Benutzer kann nur dann auf den Transaktionscode zugreifen, wenn das Keyset des Benutzers und das Keyset des LTERM-Partners, über den der Benutzer angemeldet ist, mindestens einen Keycode enthält
  - der dem Lockcode entspricht oder
  - der auch in dem in *access_list* genannte Keyset enthalten ist.

  Wirkungsdauer / Transaktionssicherung: Typ GPD ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

- Zugriffsschutz für eine TAC-Queue verändern. In der Datenstruktur *kc_tac_str* geben Sie Folgendes an:

Feldname	Bedeutung
q_read_acl[8]	In *q_read_acl* geben Sie den Namen eines existierenden Keysets an, über das die Queue vor unbefugtem Lesen und Löschen geschützt wird. Sie können den Schutz auch entfernen, indem Sie Leerzeichen angeben. In diesem Fall können alle Benutzer Nachrichten aus dieser Queue lesen und dabei löschen.
q_write_acl[8]	In *q_write_acl* geben Sie den Namen eines existierenden Keysets an, über das die Queue vor unbefugten Schreibzugriffen geschützt wird. Sie können den Schutz auch entfernen, indem Sie Leerzeichen angeben. In diesem Fall können alle Benutzer Nachrichten in diese Queue schreiben.

  Ein Benutzer kann nur dann lesend (löschend) oder schreibend auf diese TAC-Queue zugreifen, wenn das Keyset des Benutzers und das Keyset des logischen Terminals, über das der Benutzer angemeldet ist, jeweils mindestens einen Keycode enthalten, der auch in dem angegebenen Keyset enthalten ist.

  Wirkungsdauer / Transaktionssicherung: Typ GPD ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

- Speichern von Asynchron-Nachrichten in der Dead Letter Queue (TAC-Queue KDCDLETQ) festlegen. In der Datenstruktur *kc_tac_str* geben Sie Folgendes an:

Feldname	Bedeutung
dead_letter_q='Y'	Nachrichten an diesen Asynchron-TAC oder diese TAC-Queue, die nicht verarbeitet werden konnten, werden in der Dead Letter Queue gesichert, sofern sie nicht erneut zugestellt werden (Redelivery) und (bei Message-Komplexen) kein negativer Quittungs-Auftrag definiert wurde. *dead_letter_q*='Y' ist nicht erlaubt für KDCDLETQ, KDCMSGTC, alle Dialog-TACs und Asynchron-TACs mit CALL=NEXT.
dead_letter_q='N'	Nachrichten an diesen Asynchron-TAC oder diese TAC-Queue, die nicht verarbeitet werden konnten, werden nicht in der Dead Letter Queue gespeichert, sondern gelöscht. Dieser Wert muss für alle Dialog-TACS und für Asynchron-TACS mit CALL=NEXT, sowie für KDCMSGTC und KDCDLETQ angegeben werden.

Wirkungsdauer / Transaktionssicherung: Typ GPD ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

11.2.9.14 obj_type=KC_TACCLASS

Die Aktion bezieht sich auf eine TAC-Klasse der UTM-Anwendung.

Für die Modifikation der Eigenschaften einer TAC-Klasse müssen Sie im Identifikationsbereich die Nummer der TAC-Klasse übergeben (Feld *kc_name2* der Union *kc_id_area*).
Im Datenbereich müssen Sie die Datenstruktur *kc_tacclass_str* mit den neuen Eigenschaftswerten übergeben.

- *Mögliche Modifikation*

  Sie können die Anzahl der Prozesse, die gleichzeitig Aufträge für Transaktionscodes der TAC-Klasse bearbeiten dürfen, herauf- oder herabsetzen. Dazu können Sie:

  - die Anzahl der Prozesse absolut angeben (*tasks*), d.h.:

    Sie geben die Anzahl der Prozesse an, die gleichzeitig Aufträge für diese TAC-Klasse bearbeiten dürfen. Bei der absoluten Angabe ist die Prozesszahl unabhängig von der aktuell eingestellten Gesamtzahl der Prozesse, in denen das Anwendungsprogramm abläuft. Dies gilt, solange die aktuelle Gesamtzahl an Prozessen der Anwendung nicht kleiner als die für die TAC-Klasse eingestellte Prozesszahl wird. In diesem Fall wird die Prozesszahl entsprechend herabgesetzt.

  - die Anzahl der Prozesse relativ angeben (*tasks_free*), d.h.:

    Sie geben die Anzahl der Prozesse an, die mindestens für die Bearbeitung von Aufträgen für Transaktionscodes anderer TAC-Klassen frei bleiben müssen. Bei der relativen Angabe ist die Anzahl der Prozesse für diese TAC-Klasse abhängig von der aktuell eingestellten Gesamtzahl der Prozesse der Anwendung. Wird die Gesamtzahl der Prozesse herabgesetzt, dann wird implizit auch die maximale Anzahl der Prozesse herabgesetzt, die Aufträge für die TAC-Klasse bearbeiten. Analog wird beim Heraufsetzen der Gesamtzahl implizit die Prozesszahl für diese TAC-Klasse heraufgesetzt.

  Die Anzahl der Prozesse einer TAC-Klasse kann nur modifiziert werden, wenn die Anwendung ohne Prioritätensteuerung generiert ist, d.h. wenn die KDCDEF-Generierung keine TAC-PRIORITIES-Anweisung enthält.

  Für die Änderung müssen Sie Folgendes in der Struktur *kc_tacclass_str* angeben:

Feldname	Bedeutung
tasks	In *tasks* geben Sie die maximale Anzahl der Prozesse an, die gleichzeitig Aufträge für Transaktionscodes der TAC-Klasse bearbeiten dürfen. Für diese TAC-Klasse wird eine zuvor durch *tasks_free* gemachte relative Angabe ausgeschaltet. Minimalwert von *tasks*: Für Dialog-TAC-Klassen (TAC-Klassen 1-8) muss *tasks* >= '1' sein, da sonst Dialog-Vorgänge blockiert würden und die Benutzer am Terminal warten müssten, bis wieder Prozesse zugelassen werden. Für Asynchron-TAC-Klassen (Klasse 9-16) darf *tasks* >= '0' sein. Maximalwert: siehe Tabelle. Ist die Angabe für *tasks* größer als die aktuell eingestellte Gesamtzahl der Prozesse für die Anwendung, dann setzt UTM den Wert automatisch auf diese Anzahl herab.

Feldname	Bedeutung
tasks_free	In *tasks_free* geben Sie Folgendes an: ○ bei Dialog-TAC-Klassen: die Anzahl der Prozesse, die mindestens für die Bearbeitung von Aufträgen für andere TAC-Klassen freigehalten werden sollen. Wird die in *tasks_free* stehende Prozessanzahl größer als die Gesamtzahl der Prozesse, die für das Anwendungsprogramm zur Verfügung steht, dann steht dieser TAC-Klasse trotzdem weiterhin ein Prozess für die Bearbeitung ihrer Transaktionscodes zur Verfügung. ○ bei Asynchron-TAC-Klassen: die Anzahl der Prozesse, die mindestens für die Bearbeitung der Transaktionscodes anderer Asynchron-TAC-Klassen freigehalten werden soll. Wird die in *tasks_free* stehende Prozessanzahl größer als die Maximalzahl der Prozesse, die gleichzeitig für die Asynchron-Verarbeitung verwendet werden können, dann werden keine Aufträge an Transaktionscodes dieser TAC-Klasse mehr bearbeitet. Minimalwert: '0' Maximalwert: siehe Tabelle.

tasks und *tasks_free* dürfen nicht zusammen in einem KC_MODIFY_OBJECT-Aufruf angegeben werden.

Der erlaubte Maximalwert für *tasks* und *tasks_free* ist abhängig von folgenden Faktoren:

- davon, ob in der TAC-Klasse Teilprogramme mit blockierenden Aufrufen (*pgwt*='Y') ablaufen dürfen oder nicht.
- von den in der KDCDEF-Steueranweisung MAX statisch generierten Werten für TASKS, TASKS-IN-PGWT und ASYNTASKS.

Die folgende Tabelle enthält die erlaubten Maximalwerte für *tasks* und *tasks_free.* Geben Sie größere Werte an, dann wird der KC_MODIFY_OBJECT-Aufruf zurückgewiesen.

TAC-Klasse	Inhalt von pgwt	erlaubter Maximalwert für tasks	erlaubter Maximalwert für tasks_free
1 - 8 (Dialog-TACs)	'N'	TASKS *)	TASKS - 1 *)
	'Y'	TASKS-IN-PGWT *)	TASKS - 1 *)
9 - 16 (Asynchron-TACs)	'N'	ASYNTASKS *)	ASYNTASKS *)
	'Y'	der kleinere der Werte: ASYNTASKS, TASKS-IN-PGWT*)	ASYNTASKS *)

*) wie in der KDCDEF-Steueranweisung MAX statisch generiert

Wirkungsdauer / Transaktionssicherung: Typ A ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

KDCTCL ("KDCTCL - Prozess-Anzahl einer TAC-Klasse ändern")

- Die Statistikwerte „Mittlere Wartezeit der Aufträge in den Auftragswarteschlangen“

  und „Anzahl der Wartesituation“ zurücksetzen. Beide Werte können nur zusammen zurückgesetzt werden.

  Die Werte können entweder für die in der Id-Area angegebene TAC-Klasse oder für alle TAC-Klassen zurückgesetzt werden:

  - Sollen sie für alle TAC-Klassen zurückgesetzt werden, dann muss in der Id-Area binär null angegeben werden. In diesem Fall dürfen *tasks* und *tasks_free* nicht geändert werden.
  - Soll eine bestimmte TAC-Klasse modifiziert werden, können *avg_wait_time_msec* und *nr_waits* zusammen mit *tasks* und *tasks_free* angegeben werden.

  In der Datenstruktur *kc_tacclass_str* geben Sie Folgendes an:

Feldname	Bedeutung
avg_wait_time_msec[10]	enthält die mittlere Wartezeit der Aufträge in den Auftragswarteschlangen, die den Transaktionscodes dieser TAC-Klasse zugeordnet sind. Die Einheit des Werts *avg_wait_time_msec* ist Millisekunde. Wenn kein Prozess für die TAC-Klasse zur Verfügung steht, nimmt UTM Aufträge für die TAC-Klasse entgegen (mit freien Prozessen, die keine Aufträge an diese TAC-Klasse bearbeiten „dürfen“) und speichert sie in der KDCFILE zwischen. Das ist immer dann der Fall, wenn Aufträge für TAC-Klassen höherer Priorität anstehen (bei Prioritätensteuerung) bzw. (bei Prozessbegrenzung) bereits die maximal erlaubte Anzahl an Prozessen Transaktionscodes der TAC-Klasse bearbeiten (siehe *tasks*, *tasks_free*) Die Zeit zwischen Entgegennehmen eines Auftrags und dem Start seiner Bearbeitung ist die hier angezeigte Wartezeit. Diesen Wert können Sie auf '0' zurücksetzen.
nr_waits[10]	Anzahl von Wartesituationen, die für die Berechnung des Wertes *avg_wait_time_msec* berücksichtigt wurden. Diesen Wert können Sie auf '0' zurücksetzen.
nr_calls[10]	Anzahl von Teilprogrammläufen für diese TAC-Klasse.

Wirkungsdauer / Transaktionssicherung: Typ GIR ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

11.2.9.15 obj_type=KC_TPOOL

Die Aktion bezieht sich auf einen LTERM-Pool der UTM-Anwendung.

Im Identifikationsbereich müssen Sie den Namen des LTERM-Pools (LTERM-Präfix) übergeben. Dazu steht das Feld *kc_name8* der Union *kc_id_area* zur Verfügung.

Im Datenbereich müssen Sie die Datenstruktur *kc_tpool_str* mit den neuen Eigenschaftswerten übergeben.

Mögliche Modifikationen

- Sie können die Zahl der Clients, die sich gleichzeitig über diesen LTERM-Pool anschließen können, herauf- bzw. herabsetzen. D.h. Sie geben an, wieviele LTERM-Partner des LTERM-Pools zugelassen bzw. gesperrt sein sollen. Über jeden LTERM-Partner des LTERM-Pools, der nicht gesperrt ist, kann sich ein Client an die Anwendung anschließen. Die Anzahl der LTERM-Partner des LTERM-Pools wird bei der KDCDEF-Generierung festgelegt. Das ist gleichzeitig die maximale Anzahl der LTERM-Partner, die für diesen LTERM-Pool zugelassen werden kann. In der Datenstruktur *kc_tpool_str* geben Sie Folgendes an:

Feldname	Bedeutung
state='N' state_number=...	Von der Gesamtzahl der LTERM-Partner dieses LTERM-Pools (siehe *kc_tpool_str. max_number* im Abschnitt "kc_tpool_str - LTERM-Pools der Anwendung") soll die in *state_number* angegebene Anzahl gesperrt sein. Die Anzahl der für diesen LTERM-Pool zugelassenen LTERM-Partner ist dann: *max_number* - *state_number*. Soll der gesamte LTERM-Pool gesperrt werden, dann müssen Sie in *state_number* den Wert von *max_number* angeben. Wollen Sie alle LTERM-Partner des LTERM-Pools freigeben, dann geben Sie *state_number*='0' an. Minimalwert für *state_number*: '0' Maximalwert für *state_number*: die in *kc_tpool_str.max_number* zurückgelieferte Maximalzahl
state='Y' state_number=...	Von der Gesamtzahl der LTERM-Partner soll nur die in *state_number* angegebene Anzahl zugelassen sein. Sollen alle LTERM-Partner des LTERM-Pools zugelassen werden, dann müssen Sie in *state_number* den generierten Maximalwert angeben (*kc_tpool_str.max_number* im Abschnitt "kc_tpool_str - LTERM-Pools der Anwendung"). Den ganzen LTERM-Pool können Sie sperren mit *state_number*='0' Minimalwert für *state_number*: '0' Maximalwert für *state_number*: die in *kc_tpool_str.max_number* zurückgelieferte Maximalzahl

  Die Felder *state* und *state_number* müssen immer zusammen angegeben werden.

  Überschreitet die Anzahl in *state_number* die generierte Maximalzahl der LTERM-Partner, dann setzt UTM automatisch den Wert von *state_number* auf diese Maximalzahl zurück.

  Wirkungsdauer / Transaktionssicherung: Typ GP ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

Die Sperre von LTERM-Partnern des LTERM-Pool wirkt sich wie folgt aus:

- ein Verbindungsaufbauwunsch eines Clients über diesen LTERM-Pool wird von UTM abgelehnt, sobald die erlaubte Anzahl an Clients, die über diesen LTERM-Pool mit der Anwendung verbunden sind, erreicht ist (alle zugelassenen LTERM-Partner sind belegt).
- überschreitet die Anzahl der zu diesem LTERM-Pool bestehenden Verbindungen in dem Moment, in dem der Aufruf von UTM bearbeitet wird, die Anzahl der für den LTERM-Pool zugelassenen LTERM-Partner, dann bleiben zunächst alle bestehenden Verbindungen erhalten.

  Die Sperre wird nur für neue Verbindungsaufbauwünsche wirksam.

  Meldet sich ein Terminal-Benutzer mit KDCOFF BUT ab, dann kann er sich mit KDCSIGN wieder anmelden, auch wenn zu diesem Zeitpunkt noch mehr Clients über den LTERM-Pool mit der Anwendung verbunden sind als zugelassen. Dies ist möglich, weil die Verbindung in diesem Fall bestehen bleibt.

  KDCPOOL ("KDCPOOL - LTERM-Pools administrieren")

- das Zeitintervall ändern, in dem UTM nach dem Ende einer Transaktion oder nach dem Anmelden maximal auf eine Eingabe vom Client wartet. Bei Zeitüberschreitung wird die Verbindung zum Client abgebaut. In der Datenstruktur *kc_tpool_str* geben Sie Folgendes an:

Feldname	Bedeutung
idletime[5]	In *idletime* geben Sie die Zeit in Sekunden an, die UTM außerhalb einer Transaktion, d.h. nach dem Ende einer Transaktion oder nach dem Anmelden, maximal auf eine Eingabe vom Client wartet. *idletime*='0' bewirkt, dass unbegrenzt gewartet wird. Maximalwert: '32767' Minimalwert: '60 Bei Werten ungleich 0 und kleiner als 60 wird der Wert 60 verwendet.

Wirkungsdauer / Transaktionssicherung: Typ GP ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

Die Änderung des Timers wird beim nächsten Transaktionsende wirksam, aber nicht vor dem Ende des Teilprogrammlaufs (PEND), in dem der Aufruf bearbeitet wird.

11.2.9.16 obj_type=KC_USER

Die Aktion bezieht sich auf eine Benutzerkennung der UTM-Anwendung und deren Queue.

Im Identifikationsbereich ist der Name der Benutzerkennung zu übergeben (Feld *kc_name8* der Union *kc_id_area*). Im Datenbereich müssen Sie die Datenstruktur *kc_user_str* mit den neuen Eigenschaftswerten übergeben.

Mögliche Modifikationen

- eine Benutzerkennung sperren oder wieder zulassen.

  Über eine gesperrte Benutzerkennung kann sich kein Benutzer oder Client mehr bei der Anwendung anmelden. Benutzerkennungen mit Administrationsberechtigung können nicht gesperrt werden.

Feldname	Bedeutung
state='N'	Die Benutzerkennung soll gesperrt werden. Ist in dem Moment, in dem die Benutzerkennung gesperrt wird, der Benutzer bei der Anwendung angemeldet, dann wird die Verbindung zu ihm nicht unterbrochen. Die Sperre wirkt erst beim nächsten Versuch eines Benutzers oder Clients, sich über diese Benutzerkennung bei der Anwendung anzumelden. Lese- und Schreibzugriffe auf die Queue einer gesperrten Benutzerkennung sind möglich.
state='Y'	Die Benutzerkennung soll wieder freigegeben werden, d.h. existiert eine Sperre, dann soll sie aufgehoben werden.

  Wirkungsdauer / Transaktionssicherung: Typ GPD ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

- das Keyset ändern, das der Benutzerkennung zugeordnet ist. In der Datenstruktur *kc_user_str* geben Sie Folgendes an:

Feldname	Bedeutung
kset[8]	In *kset* geben Sie den Namen eines existierenden Keysets an, das die Zugriffsrechte der Benutzerkennung innerhalb der Anwendung festlegt. Der Name eines Keysets kann bis zu 8 Zeichen lang sein. Der Benutzer kann auf einen mit einem Lockcode oder einer Access Liste geschützten Service nur dann zugreifen, wenn das Keyset der Benutzerkennung **und** das Keyset des LTERM-Partners, über den sich der Benutzer an die Anwendung anschließt, einen Key-/Zugangscode enthält, der entweder mit dem Lockcode des Services oder mit mindestens einem Key der Access Liste des Services übereinstimmt. Wenn Sie die bisherige Zuordnung lösen wollen, geben Sie Leerzeichen ein.

  Wirkungsdauer / Transaktionssicherung: Typ GPD ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

- das Passwort für eine Benutzerkennung ändern oder löschen.

  Beim Ändern eines Passworts müssen Sie die beim Erzeugen der Benutzerkennung festgelegte Komplexitätsstufe und die Minimallänge des Passwortes beachten. Komplexitätsstufe und Minimallänge können Sie mit KC_GET_OBJECT (Objekttyp KC_USER) abfragen. UTM liefert die Einstellungen in den Feldern *protect_pw_compl* und *protect_pw16_lth* der Datenstruktur *kc_user_str* zurück. Im Abschnitt "kc_user_str, kc_user_fix_str, kc_user_dyn1_str bzw. kc_user_dyn2_str - Benutzerkennungen" (unter Punkt

"protect_pw_compl") ist beschrieben, welche Komplexitätsstufen es gibt und welchen Kriterien ein Passwort einer bestimmten Komplexitätsstufe genügen muss.

Passwörter löschen können Sie nur, wenn:

- die beim Erzeugen der Benutzerkennung festgelegte Minimallänge des Passworts (*protect_pw16_lth*) gleich '0' ist und
- für die Benutzerkennung keine besondere Komplexitätsstufe definiert ist (*protect_pw_compl*='0').

Ist für eine Benutzerkennung ein Passwort mit begrenzter Gültigkeitsdauer definiert (*protect_pw_time* !='0' im Abschnitt "kc_user_str, kc_user_fix_str, kc_user_dyn1_str bzw. kc_user_dyn2_str - Benutzerkennungen"), dann können Sie bei der Passwortänderung als neues Passwort nicht das alte Passwort verwenden.

In Anwendungen, die mit SIGNON GRACE=Y generiert sind, können Sie beim Ändern eines Passworts wählen (*protect_pw_time_left*):

- ob die generierte Gültigkeitsdauer für das neue Passwort wirksam sein soll (vom Zeitpunkt der Änderung an) oder
- ob das Passwort sofort ungültig sein soll und beim nächsten Anmelden durch den Benutzer sofort geändert werden muss.

Wird ein Passwort mit begrenzter Gültigkeitsdauer gelöscht, dann ist keine Gültigkeitsdauer wirksam. Wird danach ein neues Passwort vergeben, ist die Gültigkeitsdauer wieder wirksam.

Beim Ändern eines Passworts müssen Sie sowohl das neue Passwort als auch den Typ des Passworts angeben. In der Datenstruktur *kc_user_str* geben Sie Folgendes an:

Feldname	**Bedeutung**
password16	Im Feld *password16* geben Sie das neue Passwort an. Zusätzlich müssen Sie im Feld *password_type* angeben, wie UTM die Angabe in *password16* interpretieren soll. Im Feld *protect_pw_time_left* können Sie verhindern, dass in Anwendungen, die mit SIGNON GRACE=Y generiert sind, ein Passwort mit beschränkter Gültigkeitsdauer sofort ungültig wird. Ist das Passwort ungültig, wird beim ersten Anmelden die Vergabe eines neuen Passworts erzwungen. Das Passwort darf bis zu 16 Zeichen lang sein. Zur Übergabe des Passworts dient die Union *kc_pw* (siehe Abschnitt "obj_type = KC_USER").
	Sie können das Passwort entweder als Character-String oder als hexadezimale Zeichenfolge angeben. Auf Unix-, Linux- und Windows-Systemen ist eine hexadezimale Angabe jedoch nur erlaubt, wenn ein schon verschlüsseltes Passwort übergeben wird, d.h. das Feld *pw_encrypted* den Wert 'Y‘ oder ‘A‘ hat. Bei einem hexadezimalen Passwort wird jedes Halb-Byte als ein Zeichen dargestellt. Geben Sie ein Passwort an, das aus weniger als 16 Zeichen besteht, dann muss *password16* rechts mit Leerzeichen (*password_type*='C'), bzw. mit dem hexadezimalen Wert für Leerzeichen (*password_type*='X') aufgefüllt werden. Zum Löschen eines Passworts geben Sie in *password16* nur Leerzeichen an oder Sie geben in *password_type* 'N' an.

Feldname	Bedeutung
password_type	In *password_type* müssen Sie angeben, wie das Passwort in *password16* zu interpretieren ist. Folgende Angaben sind möglich: ○ 'C': Das Passwort in *password16* ist als Character-String zu interpretieren. ○ 'X': Das Passwort in *password16* ist als hexadezimales Passwort zu interpretieren. Auf Unix, Linux- oder Windows-Systemen ist dies nur erlaubt, wenn ein schon verschlüsseltes Passwort übergeben wird (*pw_encrypted* = 'Y'/'A'). ○ 'N': Kein Passwort. Im Feld *password16* darf nichts angegeben werden. Ein vorhandenes Passwort wird damit gelöscht. ○ 'R': Als Passwort wird ein zufälliges Passwort erzeugt (random). Bevor sich der so generierte Benutzer anmelden kann, muss der Administrator das Passwort explizit neu setzen.
pw_encrypted	Das Feld muss auf den Wert 'Y' oder 'A' gesetzt werden, wenn das Passwort verschlüsselt übergeben wird. Dies kann z.B. dann der Fall sein, wenn das verschlüsselte Passwort aus der Meldung K159 in einer Standby-Anwendung stammt. ○ 'N': Das Passwort wird in unverschlüsselter Form übergeben (Standard). ○ 'Y'/'A': Das Password wird verschlüsselt übergeben. Es wird keine Komplexitätsprüfung vorgenommen.

Wirkungsdauer / Transaktionssicherung: Typ GPD ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

Feldname	Bedeutung
protect_pw_time_left	nur relevant in Anwendungen, die mit SIGNON GRACE=Y generiert sind und für Benutzerkennungen, für die eine begrenzte Gültigkeitsdauer des Passwortes generiert sind. In *protect_pw_time_left* können Sie angeben, ob für das neue Passwort die generierte Gültigkeitsdauer wirksam sein soll: Geben Sie *protect_pw_time_left*='-1' (rechts- oder linksbündig) an, dann ist die generierte Gültigkeitsdauer (vom Zeitpunkt der Änderung an) für das neue Passwort wirksam. *protect_pw_time_left*='-1' ist nur zusammen mit *password16* und *password_type* wirksam. *protect_pw_time_left*='-1' ohne Angabe eines Passworts wird ignoriert. Geben Sie in *protect_pw_time_left* nichts an, dann ist das neue Passwort wegen Ablauf der Gültigkeitsdauer sofort ungültig. Der Benutzer muss beim nächsten Anmelden sein Passwort ändern. Ein anderer Wert als '-1' wird zurückgewiesen.

- Schreib-, Lese- und Löschberechtigung für eine USER-Queue verändern. In der Datenstruktur *kc_user_str* geben Sie Folgendes an:

Feldname	Bedeutung
q_read_acl[8]	In *q_read_acl* geben Sie den Namen eines existierenden Keysets an, über das die Queue vor fremden Benutzern geschützt wird, die lesend und dabei löschend auf die Queue zugreifen wollen. Sie können den Schutz entfernen, indem Sie Leerzeichen angeben. In diesem Fall können alle Benutzer Nachrichten aus dieser Queue lesen und dabei löschen.
q_write_acl[8]	In *q_write_acl* geben Sie den Namen eines existierenden Keysets an, über das die Queue vor fremden Benutzern geschützt wird, die schreibend auf die Queue zugreifen wollen. Sie können den Schutz entfernen, indem Sie Leerzeichen angeben. In diesem Fall können alle Benutzer Nachrichten in diese Queue schreiben.

  Ein fremder Benutzer (!=*us_name*) kann nur dann lesend (löschend) oder schreibend auf die USER-Queue zugreifen, wenn sowohl das Keyset seiner Benutzerkennung als auch das Keyset des LTERM-Partners, über den der Benutzer angemeldet ist, jeweils mindestens einen Keycode des Keysets *q_read_acl* bzw. *q_write_acl* enthält.

  Wirkungsdauer / Transaktionssicherung: Typ GPD ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

- Nur auf BS2000-Systemen:
  der Benutzerkennung ein neues Startformat zuordnen.

  Jeder Benutzerkennung können Sie ein spezifisches Startformat zuordnen. Dieses Startformat wird nach jedem erfolgreichen Anmelden automatisch ausgegeben, wenn kein offener Vorgang für diese Benutzerkennung existiert. Zum Ändern des Startformats müssen Sie immer Formatname und Formatattribut angeben.

  Voraussetzung für das Zuweisen eines Startformats ist, dass ein Formatierungssystem generiert ist (KDCDEF-Anweisung FORMSYS). Ist das Startformat ein #Format, dann muss zusätzlich ein Anmelde-Vorgang generiert sein.

Feldname	Bedeutung	
format_attr	Formatkennzeichen des neuen Startformats:	
	'A'	für das Formatattribut ATTR. Der Formatname an der Programmschnittstelle KDCS ist *+format_name*.
	'N'	für das Formatattribut NOATTR. Der Formatname an der Programmschnittstelle KDCS ist **format_name*.
	'E'	für das Formatattribut EXTEND. Der Formatname an der Programmschnittstelle KDCS ist *#format_name*.
	Die Bedeutung der Formatattribute ist im Abschnitt "kc_user_str, kc_user_fix_str, kc_user_dyn1_str bzw. kc_user_dyn2_str - Benutzerkennungen" beschrieben.	

Feldname	Bedeutung
format_name[7]	Name des Startformats. Der Name kann bis zu 7 Zeichen lang sein und darf nur alphanumerische Zeichen enthalten.

Wollen Sie das Startformat einer Benutzerkennung löschen, dann müssen Sie in *format_attr* und *format_name* Leerzeichen angeben.

Wirkungsdauer / Transaktionssicherung: Typ GPD ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

- den BCAM-Trace für diese Benutzerkennung ein- oder ausschalten.

  Voraussetzung für das USER-spezifische Einschalten ist, dass der BCAM-Trace nicht allgemein für alle Verbindungen eingeschaltet ist. D.h. der Trace ist entweder ganz ausgeschaltet oder nur für einige ausgewählte LTERM- und /LPAP-Partner oder USER explizit eingeschaltet. In der Datenstruktur *kc_user_str* geben Sie Folgendes an:

Feldname	Bedeutung
bcam_trace='Y'	Der BCAM-Trace wird für diesen USER explizit eingeschaltet. Dies ist nur möglich, ○ wenn der BCAM-Trace für alle Verbindungen ausgeschaltet ist (siehe *kc_diag_and_account_par_str*) oder ○ wenn der BCAM-Trace bereits für einzelne USER eingeschaltet ist.
bcam_trace='N'	Der BCAM-Trace wird für diesen USER ausgeschaltet.

Wirkungsdauer / Transaktionssicherung: Typ GIR ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

Einige Modifikationen sind auch mit KDCUSER ("KDCUSER - Benutzereigenschaften ändern") oder mit KDCDIAG ("Diagnosehilfen ein- und ausschalten") durchführbar.

11.2.9.17 obj_type = KC_CLUSTER_CURR_PAR

In einer UTM-Cluster-Anwendung (Unix-, Linux- und Windows-Systeme) sollen die Statistikwerte des Cluster Pagepools zurückgesetzt werden.

Über den Datenbereich müssen Sie die Datenstruktur *kc_cluster_curr_par_str* legen.

Mögliche Modifikationen

Der folgenden Tabelle können Sie entnehmen, welche Werte Sie zurücksetzen können.

Feldname	Bedeutung
max_cpgpool_size='0'	Maximale Belegung des Cluster Pagepools. Der Zähler wird auf 0 zurückgesetzt.
avg_cpgpool_size='0'	Durchschnittliche Belegung des Cluster Pagepools. Der Zähler wird auf 0 zurückgesetzt.

Beim Zurücksetzen eines der beiden Werte wird implizit auch der andere Wert zurückgesetzt.

Wirkungsdauer / Transaktionssicherung: Typ GID ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

Ohne explizites Zurücksetzen gelten die Werte über den Lauf der gesamten Cluster-Anwendung hinaus und werden erst bei Vergrößerung des Cluster Pagepools und beim Erzeugen der UTM-Cluster-Dateien per KDCDEF zurückgesetzt.

11.2.9.18 obj_type=KC_CLUSTER_PAR

Es sollen die Einstellungen für die Ringüberwachung der Knoten-Anwendungen einer UTM-Cluster-Anwendung (Unix-, Linux- und Windows-Systeme) und/oder die Einstellungen für den Zugriff der Knoten-Anwendungen auf die Cluster-Konfigurationsdatei und das Administrations-Journal der UTM-Cluster-Anwendung (Unix-, Linux- und Windows-Systeme) geändert werden.

Über den Datenbereich müssen Sie dazu die Datenstruktur *kc_cluster_par_str* mit den neuen Eigenschaftswerten legen.

Mögliche Modifikation

Der folgenden Tabelle können Sie entnehmen, welche Einstellungen Sie ändern können.

Feldname	Bedeutung
check_alive_timer_sec	In einer UTM-Cluster-Anwendung wird jede Knoten-Anwendung durch eine andere Knoten-Anwendung überwacht (Ringüberwachung), d.h. jede Knoten-Anwendung überprüft die Verfügbarkeit einer anderen Knoten-Anwendung und wird selbst von einer Knoten-Anwendung überwacht. Dazu schickt die überwachende Knoten-Anwendung in bestimmten Zeitintervallen (*check_alive_timer_sec*) Nachrichten an die zu überwachende Knoten-Anwendung. Ist diese verfügbar, quittiert diese die Nachricht. *check_alive_timer_sec* gibt den Zeitabstand in Sekunden an, in dem Überwachungs-Nachrichten an die zu überwachende Knoten-Anwendung geschickt werden. Dieser Timer wird von openUTM auch verwendet, um periodisch auf die Cluster-Konfigurationsdatei und das Administrations-Journal zuzugreifen und diese Dateien auf evtl. Aktualisierungen zu prüfen. Minimalwert: '30' Maximalwert: '3600'
communication_retry	*communication_retry* gibt an, wie oft eine Knoten-Anwendung den Versuch wiederholt, eine Überwachungs-Nachricht zu senden, wenn die zu überwachende Knoten-Anwendung nicht innerhalb der festgesetzten Zeit antwortet. Ist für *communication_retry* ein Wert größer Null gesetzt, dann wird von einem Ausfall der anderen Knoten-Anwendung erst ausgegangen, wenn eine Antwort auf die Überwachungs-Nachricht auch nach dem letzten Wiederholversuch ausbleibt. Minimalwert: '0' Maximalwert:'10'

Feldname	Bedeutung
communication_reply_timer_sec	*communication_reply_timer_sec* gibt die Zeit in Sekunden an, die eine Knoten-Anwendung nach dem Senden einer Überwachungs-Nachricht maximal auf Antwort wartet. Antwortet die zu überwachende Knoten-Anwendung nicht in der festgesetzten Zeit, dann wird ihr Ausfall (abnormales Anwendungsende) angenommen und die in *failure_cmd* definierte Kommandofolge ausgeführt (z. B. ein Neustart). Minimalwert: '1' Maximalwert: '60'
restart_timer_sec	Zeit in Sekunden, die eine Knoten-Anwendung nach einem Ausfall maximal für einen Warm-Start benötigt. Bei Angabe des Wertes 0 wird der Neu-Start einer ausgefallenen Anwendung nicht zeitüberwacht. Minimalwert: 0, d.h. keine Überwachung eines Anwendungs-Restarts Maximalwert: 3600
file_lock_timer_sec file_lock_retry	*file_lock_timer_sec* ist die Zeit in Sekunden, die eine Knoten-Anwendung maximal auf die Zuteilung einer Sperre für den Zugriff auf die Cluster-Konfigurationsdatei oder das Cluster-Administrations-Journal wartet. *file_lock_retry* gibt an, wie oft eine Knoten-Anwendung die Anforderung einer Sperre für die Cluster-Konfigurationsdatei oder das Cluster-Administrations-Journal wiederholt, wenn die Sperre nicht in der in *file_lock_timer_sec* vorgegebenen Zeit zugeteilt werden konnte. Hinweis: Wählen Sie diese Werte nicht zu klein, da ein Timeout beim Zugriff auf die Cluster-Konfigurationsdatei zum abnormalen Anwendungsende führen kann. *file_lock_timer_sec*: Minimalwert:'10' Maximalwert:'60' *file_lock_retry*: Minimalwert:'1' Maximalwert:'10'
deadlock_prevention='N'	UTM führt für die Datenbereiche GSSB, TLS und ULS keine zusätzlichen Prüfungen zur Deadlock-Vermeidung durch. Kommt es zu einem Deadlock auf diesen Datenbereichen, dann wird dieser über einen Timeout aufgelöst.
deadlock_prevention='Y'	UTM führt für die Datenbereiche GSSB, TLS und ULS zusätzliche Prüfungen zur Deadlock-Vermeidung durch. Es wird empfohlen, diesen Parameter im Produktivbetrieb nur dann auf 'Y' zu setzen, wenn es häufig zu Timeouts beim Zugriff auf diese Datenbereiche kommt.

Wirkungsdauer / Transaktionssicherung: Typ GID ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern").

11.2.9.19 obj_type = KC_CURR_PAR

Es sollen Zähler für Anwendungs-spezifische Statistikwerte zurückgesetzt werden. Über den Datenbereich müssen Sie dazu die Datenstruktur *kc_curr_par_str* legen.

Außerdem können Sie die Datenkomprimierung ein- oder ausschalten, siehe "Datenkomprimierung ein-/ausschalten".

Mögliche Modifikationen

Alle im Folgenden aufgelisteten Zähler können in einem Aufruf zurückgesetzt werden. Zum Zurücksetzen eines Zählers müssen Sie im entsprechenden Feld, falls nicht anders vermerkt, den Wert '0' an UTM übergeben.

Folgende Zählerstände bzw. Statistikwerte können Sie zurücksetzen:

Feldname	**Bedeutung**
term_input_msgs='0'	Anzahl aller Nachrichten, die die Anwendung seit dem letzten Zurücksetzen von Clients oder Partner-Anwendungen empfangen hat. Der Zähler wird auf 0 zurückgesetzt.
term_output_msgs='0'	Anzahl aller Nachrichten, die die Anwendung seit dem letzten Zurücksetzen an Clients, Drucker oder Partner-Anwendungen gesendet hat. Der Zähler wird auf 0 zurückgesetzt.
max_dial_ta_per_100sec='0'	Maximale Anzahl der Dialog-Transaktionen, die innerhalb eines Zeitintervalls von 100 Sekunden ausgeführt wurden. Der Zähler wird auf 0 zurückgesetzt ("kc_curr_par_str - Aktuelle Werte der Anwendungsparameter").
max_asyn_ta_per_100sec='0'	Maximale Anzahl der Asynchron-Transaktionen, die innerhalb eines Zeitintervalls von 100 Sekunden ausgeführt wurden. Der Zähler wird auf 0 zurückgesetzt ("kc_curr_par_str - Aktuelle Werte der Anwendungsparameter").
max_dial_step_per_100sec='0'	Maximale Anzahl der Dialog-Schritte, die innerhalb eines Zeitintervalls von 100 Sekunden ausgeführt wurden. Der Zähler wird auf 0 zurückgesetzt ("kc_curr_par_str - Aktuelle Werte der Anwendungsparameter").
max_pool_size='0'	Maximale Belegung des Pagepools in Prozent seit dem letzten Zurücksetzen. Der Zähler wird auf 0 zurückgesetzt. Bei einem Zurücksetzen dieses Werts wird implizit auch der Wert *avg_pool_size* auf 0 zurückgesetzt.
avg_pool_size='0'	Mittlere Belegung des Pagepools in Prozent seit dem letzten Zurücksetzen des Zählers. Der Zähler wird auf 0 zurückgesetzt. Bei einem Zurücksetzen dieses Werts wird implizit auch der Wert *max_pool_size* auf 0 zurückgesetzt.

Feldname	Bedeutung
cache_hit_rate='0'	Trefferquote bei der Suche einer Seite im Cache-Speicher seit dem letzten Zurücksetzen des Zählers (Angabe in Prozent). Der Zähler wird auf 0 zurückgesetzt. Bei einem Zurücksetzen dieses Werts werden implizit auch die Werte *cache_wait_buffer*, *nr_cache_rqs* und *nr_cache_searches* auf 0 zurückgesetzt.
cache_wait_buffer='0'	Prozentsatz der Anforderungen von Puffern im Cache, die zu einer Wartezeit geführt haben. Der Zähler wird auf 0 zurückgesetzt. Bei einem Zurücksetzen dieses Werts werden implizit auch die Werte *cache_hit_rate* , *nr_cache_rqs* und *nr_cache_searches* auf 0 zurückgesetzt.
abterm_services='0'	Anzahl der abnormal beendeten Vorgänge seit dem letzten Zurücksetzen. Der Zähler wird auf 0 zurückgesetzt.
deadlocks='0'	Anzahl der erkannten und aufgelösten Deadlocks von UTM-Betriebsmitteln seit dem letzten Zurücksetzen. Der Zähler wird auf 0 zurückgesetzt.
periodic_writes='0'	Anzahl der Periodic Writes seit dem letzten Zurücksetzen (periodic write = Sicherung der gesamten sicherungsrelevanten Verwaltungsdaten der UTM-Anwendung). Der Zähler wird auf 0 zurückgesetzt.
pages_pwrite='0'	Anzahl der UTM-Seiten, die bei einem periodic write im Mittel gesichert wurden. Der Zähler wird auf 0 zurückgesetzt.
logfile_writes='0'	Anzahl der Anforderungen, Protokollsätze auf die Benutzer-Protokolldatei (USLOG) zu schreiben. Der Zähler wird auf 0 zurückgesetzt.
maximum_jr='0'	Nur bei verteilter Verarbeitung: Maximale Anzahl der in der lokalen Anwendung gleichzeitig adressierten fernen Auftragnehmer-Vorgänge relativ zum Generierungswert MAXJR (siehe *kc_utmd_par_str* im Abschnitt "kc_utmd_par_str - Parameter für die verteilte Verarbeitung"). Die Angabe erfolgt in Prozent. Der Zähler wird auf den Wert von *curr_jr* zurückgesetzt ("kc_curr_par_str - Aktuelle Werte der Anwendungsparameter").
max_load='0'	*max_load* bezeichnet die maximale Auslastung der UTM-Anwendung in Prozent, die seit Anwendungsstart oder dem letzten Zurücksetzen registriert wurde. Der Wert wird auf den Wert von *curr_load* zurückgesetzt (siehe "kc_curr_par_str - Aktuelle Werte der Anwendungsparameter").

Feldname	Bedeutung
max_wait_resources='0'	*max_wait_resources* bezeichnet die maximale Konfliktrate für Locks auf Anwenderdaten über den Anwendungslauf. Der Wert wird in Promille angegeben. Der Zähler wird auf 0 zurückgesetzt. Bei einem Zurücksetzen dieses Werts wird implizit auch die Werte *max_wait_system_resources, nr_res_rqs_for_max* und *nr_sys_res_rqs_for_max* auf 0 zurückgesetzt.
max_wait_system_resources='0'	*max_wait_system_resources* bezeichnet die maximale Konfliktrate für Anforderungen an System-Ressourcen (Systemlocks) über den Anwendungslauf. Der Wert wird in Promille angegeben. Der Zähler wird auf 0 zurückgesetzt. Bei einem Zurücksetzen dieses Werts wird implizit auch die Werte *max_wait_resources, nr_res_rqs_for_max* und *nr_sys_res_rqs_for_max* auf 0 zurückgesetzt.
nr_cache_rqs='0'	Anzahl von Pufferanforderungen, die für die Berechnung des Werts *cache_wait_buffer* berücksichtigt wurden. Der Zähler wird auf 0 zurückgesetzt. Bei einem Zurücksetzen dieses Werts werden implizit auch die Werte *cache_hit_rate, cache_wait_buffer* und *nr_cache_searches* auf 0 zurückgesetzt.
nr_cache_searches='0'	Anzahl der Suchvorgänge nach UTM-Seiten im Cache, die für die Berechnung des Wertes *cache_hit_rate* berücksichtigt wurden. Der Zähler wird auf 0 zurückgesetzt. Bei einem Zurücksetzen dieses Werts werden implizit auch die Werte *cache_hit_rate, cache_wait_buffer* und *nr_cache_rqs* auf 0 zurückgesetzt.
nr_res_rqs_for_max='0'	Anzahl der Anforderungen an Transaktions-Ressourcen in dem 100 Sekunden Intervall, in dem die maximale Konfliktrate *max_wait_resources* erreicht wurde. Der Zähler wird auf 0 zurückgesetzt. Bei einem Zurücksetzen dieses Werts werden implizit auch die Werte *max_wait_resources, max_wait_system_resources* und *nr_sys_res_rqs_for_max* auf 0 zurückgesetzt.
nr_sys_res_rqs_for_max='0'	Anzahl der Anforderungen an System-Ressourcen in dem 100 Sekunden Intervall, in dem die maximale Konfliktrate *max_wait_system_resources* erreicht wurde. Der Zähler wird auf 0 zurückgesetzt. Bei einem Zurücksetzen dieses Werts werden implizit auch die Werte *max_wait_resources, max_wait_system_resources* und *nr_res_rqs_for_max* auf 0 zurückgesetzt.

Feldname	Bedeutung
avg_saved_pgs_by_compr='0'	Durchschnittswert der pro Datenkomprimierung eingesparten UTM-Seiten. Der Zähler wird auf 0 zurückgesetzt.

Wirkungsdauer / Transaktionssicherung: Typ GIR ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

> Wenn Sie die obigen Statistikwerte selbst zurücksetzen wollen, dann empfiehlt es sich, bei der KDCDEF-Generierung MAX STATISTICS-MSG =NONE zu setzen. Damit verhindern Sie, dass UTM die Zählerstände stündlich auf 0 zurücksetzt und die Statistikmeldung K081 erzeugt.

Datenkomprimierung ein-/ausschalten

Feldname	Bedeutung
data_compression='Y'	Die Datenkomprimierung wird eingeschaltet. Dazu muss die Datenkomprimierung per UTM-Generierung erlaubt sein, siehe openUTM-Handbuch „Anwendungen generieren“, MAX DATA-COMPRESSION=
data_compression='N'	Die Datenkomprimierung wird ausgeschaltet.

Wirkungsdauer / Transaktionssicherung: Typ GID ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

11.2.9.20 obj_type = KC_DIAG_AND_ACCOUNT_PAR

Es sollen Diagnosefunktionen ein- bzw. ausgeschaltet werden. Über den Datenbereich müssen Sie die Datenstruktur *kc_diag_and_account_par_str* legen.

Mögliche Modifikationen

- Die ADMI-Tracefunktion ein- oder ausschalten. Die ADMI-Tracefunktion protokolliert alle Aufrufe der Programmschnittstelle KDCADMI.

Feldname	Bedeutung
admi_trace='Y'	Die ADMI-Tracefunktion wird eingeschaltet.
admi_trace='N'	Die ADMI-Tracefunktion wird ausgeschaltet. Alle ADMI-Trace-Dateien werden geschlossen und können ausgewertet werden. Siehe dazu auch das openUTM-Handbuch „Meldungen, Test und Diagnose“.

  Wirkungsdauer / Transaktionssicherung: Typ IR ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

  Beim Start der Anwendung kann der Trace auch über Startparameter eingeschaltet werden, siehe openUTM-Handbuch „Einsatz von UTM-Anwendungen“. Dort sind auch die Namen der Trace-Dateien beschrieben.

- BCAM-Trace für alle Verbindungen der Anwendung ein- oder ausschalten. D.h. für alle
  - LTERM-, LPAP-Partner
  - USER
  - MUX-Verbindungen (nur auf BS2000-Systemen)

  Der BCAM-Trace zeichnet alle Verbindungs-spezifischen Ereignisse auf.

Feldname	Bedeutung
bcam_trace='Y'	Die BTRACE-Funktion wird für alle Verbindungen eingeschaltet. Beim Einschalten der BTRACE-Funktion erzeugt jeder Prozess der Anwendung seine eigene Trace-Datei, in der er die Verbindungs-spezifischen Ereignisse aufzeichnet.
bcam_trace='N'	Die BTRACE-Funktion wird für alle Verbindungen ausgeschaltet, auch wenn sie vorher nur LTERM-, LPAP- bzw. MUX- oder USER-spezifisch eingeschaltet war. Wird die BTRACE-Funktion ausgeschaltet (für alle LTERM-, LPAP-, MUX-Partner und USER), werden die Trace-Dateien geschlossen und können danach ausgewertet werden. Inhalt und Auswertung der Trace-Dateien sind im openUTM-Handbuch „Meldungen, Test und Diagnose“ beschrieben.

  Sie können den BCAM-Trace auch LTERM-, LPAP-, MUX- oder USER-spezifisch ein- bzw. ausschalten. Dazu verwenden Sie die Objekttypen KC_LTERM ("obj_type=KC_LTERM"), KC_LPAP ("obj_type=KC_LPAP"), KC_MUX ("obj_type=KC_MUX (BS2000-Systeme)") oder KC_USER ("obj_type=KC_USER").

  Wirkungsdauer / Transaktionssicherung: Typ GIR ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

  Beim Start der Anwendung kann der BCAM-Trace auch über Startparameter eingeschaltet werden.

- Die CPI-C-Tracefunktion steuern. Die CPI-C-Tracefunktion protokolliert die Aufrufe der X/Open-Schnittstelle CPI-C.

Feldname	Bedeutung
cpic_trace='T'	Die CPI-C-Tracefunktion wird mit Level TRACE eingeschaltet. Zu jedem CPI-C-Funktionsaufruf wird der Inhalt der Input- und Output-Parameter ausgegeben. Von den Datenpuffern werden nur die ersten 16 Byte ausgegeben. Die Returncodes der KDCS-Aufrufe, auf die die CPI-C-Aufrufe abgebildet werden, werden ausgegeben.
cpic_trace='B'	Die CPI-C-Tracefunktion wird mit Level BUFFER eingeschaltet. Dieser Trace-Level umfasst den Level TRACE, die Datenpuffer werden jedoch in voller Länge protokolliert.
cpic_trace='D'	Die CPI-C-Tracefunktion wird mit Level DUMP eingeschaltet. Dieser Trace-Level umfasst den Level TRACE, zusätzlich werden Diagnose-Informationen in die Trace-Datei geschrieben.
cpic_trace='A'	Die CPI-C-Tracefunktion wird mit Level ALL eingeschaltet. Dieser Trace-Level umfasst die Level BUFFER, DUMP und TRACE.
cpic_trace='N'	Die CPI-C-Tracefunktion wird ausgeschaltet (OFF). Alle CPI-C-Trace-Dateien werden geschlossen und können ausgewertet werden. Siehe dazu auch das openUTM-Handbuch „Anwendungen erstellen mit X/Open-Schnittstellen“.

Wirkungsdauer / Transaktionssicherung: Typ IR ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

Beim Start der Anwendung kann der CPI-C-Trace auch über Startparameter eingeschaltet werden, siehe openUTM-Handbuch „Einsatz von UTM-Anwendungen“. Dort sind auch die Namen der Trace-Dateien beschrieben.

- die OSI-Tracefunktionen für alle OSI-Verbindungen der Anwendung ein- oder ausschalten.

  Die OSI-Tracefunktionen zeichnen alle Ereignisse auf, die bei der verteilten Verarbeitung über OSI TP auftreten. Die Aufzeichnungen können auf bestimmte Record-Typen, d.h. auf Ereignisse bestimmter Komponenten, beschränkt werden.

  Das Ausschalten der Protokollierung für einzelne Record-Typen ist nicht möglich. Soll der Trace für einzelne Record-Typen ausgeschaltet werden, dann muss er erst ganz ausgeschaltet (*osi_trace*='N') und dann für die Record-Typen, die noch mitprotokolliert werden sollen, wieder eingeschaltet werden (entsprechende Angaben in *osi_trace_records*).

Feldname	**Bedeutung**
osi_trace='Y'	Die OSI-Tracefunktion wird für alle Record-Typen eingeschaltet. Beim Einschalten der OSI-Tracefunktion erzeugt jeder Prozess der Anwendung seine eigene Trace-Datei.
osi_trace='N'	Der OSI-Trace wird für alle Record-Typen ausgeschaltet. Alle OSI-Trace-Dateien werden geschlossen und können ausgewertet werden. Siehe dazu auch das openUTM-Handbuch „Meldungen, Test und Diagnose“.
osi_trace_records[5]	OSI-Tracefunktion für bestimmte Record-Typen einschalten. Zum Einschalten des OSI-Trace ist dann keine weitere Angabe im Feld *osi_trace* nötig. Jedes Feldelement von *osi_trace_records* repräsentiert einen Record-Typ: 1. Feld den Record-Typ „SPI“ 2. Feld den Record-Typ „INT“ 3. Feld den Record-Typ „OSS“ 4. Feld den Record-Typ „SERV“ 5. Feld den Record-Typ „PROT“ Die Bedeutung der Record-Typen ist im Abschnitt "kc_diag_and_account_par_str - Diagnose- und Accounting-Parameter" zusammengefasst. Zum Einschalten der Tracefunktionen für bestimmte Record-Typen geben Sie in den entsprechenden Feldelementen 'Y' an. Durch den Aufruf wird die Protokollierung für die angegebenen Record-Typen zusätzlich zu einer eventuell zuvor bestehenden Protokollierung eingeschaltet.

  Wirkungsdauer / Transaktionssicherung: Typ GIR ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

  Beim Start der Anwendung kann der Trace auch über Startparameter eingeschaltet werden.

- Die TX-Tracefunktion steuern. Die TX-Tracefunktion protokolliert die Aufrufe der X/Open-Schnittstelle TX.

Feldname	Bedeutung
tx_trace='E'	Die TX-Tracefunktion wird mit Level ERROR eingeschaltet. Es werden nur Fehler protokolliert.
tx_trace='I'	Die TX-Tracefunktion wird mit Level INTERFACE eingeschaltet. Der Level INTERFACE umfasst den Level ERROR, zusätzlich werden alle TX-Aufrufe protokolliert.
tx_trace='F'	Die TX-Tracefunktion wird mit Level FULL eingeschaltet. Der Level FULL umfasst den Level INTERFACE, zusätzlich werden alle KDCS-Aufrufe, auf die die TX-Aufrufe abgebildet werden, protokolliert.
tx_trace='D'	Die TX-Tracefunktion wird mit Level DEBUG eingeschaltet. Der Level DEBUG umfasst den Level FULL, zusätzlich werden Diagnose-Informationen protokolliert..
tx_trace='N'	Die TX-Tracefunktion wird ausgeschaltet. Alle TX-Trace-Dateien werden geschlossen und können ausgewertet werden. Siehe dazu auch das openUTM-Handbuch „Anwendungen erstellen mit X/Open-Schnittstellen“.

Wirkungsdauer / Transaktionssicherung: Typ IR ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

Beim Start der Anwendung kann der TX-Trace auch über Startparameter eingeschaltet werden, siehe openUTM-Handbuch „Einsatz von UTM-Anwendungen“. Dort sind auch die Namen der Trace-Dateien beschrieben.

- Die XATMI-Tracefunktion steuern. Die XATMI-Tracefunktion protokolliert die Aufrufe der X/Open-Schnittstelle XATMI.

Feldname	Bedeutung
xatmi_trace='E'	Die XATMI-Tracefunktion wird mit Level ERROR eingeschaltet. Es werden nur Fehler protokolliert.
xatmi_trace='I'	Die XATMI-Tracefunktion wird mit Level INTERFACE eingeschaltet. Der Level INTERFACE umfasst den Level ERROR, zusätzlich werden alle XATMI-Aufrufe protokolliert.
xatmi_trace='F'	Die XATMI-Tracefunktion wird mit Level FULL eingeschaltet. Der Level FULL umfasst den Level INTERFACE, zusätzlich werden alle KDCS-Aufrufe, auf die die XATMI-Aufrufe abgebildet werden, protokolliert.
xatmi_trace='D'	Die XATMI-Tracefunktion wird mit Level DEBUG eingeschaltet. Der Level DEBUG umfasst den Level FULL, zusätzlich werden Diagnose-Informationen protokolliert..
xatmi_trace='N'	Die XATMI-Tracefunktion wird ausgeschaltet. Alle XATMI-Trace-Dateien werden geschlossen und können ausgewertet werden. Siehe dazu auch das openUTM-Handbuch „Anwendungen erstellen mit X/Open-Schnittstellen“.

Wirkungsdauer / Transaktionssicherung: Typ IR ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

Beim Start der Anwendung kann der XATMI-Trace auch über Startparameter eingeschaltet werden, siehe openUTM-Handbuch „Einsatz von UTM-Anwendungen“. Dort sind auch die Namen der Trace-Dateien beschrieben.

- den Testmodus für die Anwendung ein- und ausschalten.

  Der Testmodus sollte nur zum Erstellen von Diagnoseunterlagen eingeschaltet werden. Im Testmodus laufen zusätzlich UTM-interne Routinen zur Plausibilitätsprüfung ab und es werden interne TRACE-Informationen aufgezeichnet.

Feldname	**Bedeutung**
testmode='Y'	Testmodus wird eingeschaltet (ON).
testmode='N'	Testmodus wird ausgeschaltet (OFF).

  Wirkungsdauer / Transaktionssicherung: Typ GIR ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

  Beim Start der Anwendung kann der Testmodus auch über Startparameter eingeschaltet werden.

- Diagnose-Dump bei festgelegten Meldungen/Ereignissen erzeugen.

  Sie können ein Ereignis definieren, bei dessen Auftreten UTM einen Diagnose-Dump erzeugt, der ein Ereignis-abhängiges Kennzeichen erhält. Voraussetzung dafür ist der eingeschaltete Testmodus (*testmode*='Y'). Das Einschalten des Testmodus und Definieren des Ereignisses kann in einem KC_MODIFY_OBJECT-Aufruf erfolgen. Das Ereignis können Sie auch definieren, wenn der Testmodus nicht eingeschaltet ist. Der Diagnose-Dump wird bei Auftreten des Ereignisses jedoch nur geschrieben, wenn der Testmodus eingeschaltet ist.

  Folgende Ereignisse können Sie angeben:

  - die Ausgabe einer bestimmten K- oder P-Meldung, ggf. abhängig von den Inserts der Meldung
  - das Auftreten eines bestimmten KDCS-Returncodes (KCRCCC oder KCRCDC) in einem Teilprogrammlauf
  - das Auftreten eines bestimmten SIGN-Status beim Anmelden eines Benutzers

  Die Ereignisse werden in *kc_diag_and_accout_par_str* in der Datenstruktur *kc_dump_event_str* festgelegt, die zusätzlich zu den Feldern *event_type* und *event* die Datenstruktur *kc_insert_str* enthält.

  In *kc_insert_str* werden ggf. die Meldungs-Inserts definiert, die das Erzeugen des Dumps weiter einschränken. Sie können bis zu drei Inserts angeben. Ein Dump wird dann nur gezogenen, wenn alle in kc_insert_str angegebenen Kriterien für die Meldungs-Inserts zutreffen.

  Datenstruktur kc_dump_event_str

Feldname	**Bedeutung**
event_type[4]	Typ des Ereignisses, für das ein UTM-Dump gezogen werden soll:
	MSG: K- oder P-Meldung RCDC: inkompatibler Returncode RCCC: kompatibler Returncode SIGN: SIGNON-Statuscode NONE: Explizites Ausschalten eines einzelnen Ereignisses
event[4]	Meldungsnummer, KDCS-Returncode (CC oder DC) oder SIGNON-Statuscode, abhängig vom *event_type*

  Datenstruktur kc_insert_str

Feldname	**Bedeutung**
value[64]	*value* kann in Abhängigkeit von *value_type* wie folgt angegeben werden:
	value_type=N: numerisch, ganze Zahlen von 0 bis 2^{31}-1 *value_type*=C: alphanumerisch, maximal 32 Zeichen *value_type*=X: hexadezimal, maximal 64 Zeichen

	UTM stellt die Zeichenfolge in einer Union des Typs *kc_value* dar:
	`union kc_value`
	`char x[64];`
	`char c[32];`

Feldname	Bedeutung
value_type	*value_type* legt fest wie der Inhalt des Feldes *value* zu interpretieren ist:
	N: numerisch C: alphanumerisch X: hexadezimal
comp[2]	legt fest, ob auf Gleichheit oder Ungleichheit geprüft werden soll. Mögliche Werte sind EQ (Gleichheit) oder NE (Ungleichheit)

Bei den Meldungen K023, K043, K061 oder K062 erzeugt UTM nur einmal einen UTM-Dump, und zwar beim nächsten Auftreten der Meldung. Die Funktion Message-Dump wird danach automatisch ausgeschaltet.

Bei allen anderen UTM-Meldungsnummern wird bei jedem Auftreten des angegebenen Ereignisses ein UTM-Dump erzeugt, und zwar solange, bis das Ereignis explizit zurückgesetzt wird.

Bei KDCS-Returncodes bzw. SIGNON-Statuscodes wird die Funktion nach dem Erzeugen des Message-Dump automatisch ausgeschaltet.

Wirkungsdauer / Transaktionssicherung: Typ GIR ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

KDCDIAG ("KDCDIAG - Diagnosehilfen ein- und ausschalten")

- Abrechnungs- und Kalkulationsphase des UTM-Accounting ein- bzw. ausschalten.

  Zum UTM-Accounting siehe auch openUTM-Handbuch „Anwendungen generieren“ und openUTM-Handbuch „Einsatz von UTM-Anwendungen“.

Feldname	**Bedeutung**
account='Y'	Nur auf auf BS2000-Systemen: Abrechnungsphase des Accounting einschalten (ON). Nach einem Ausfall des BS2000-Accounting ist das UTM-Accounting immer ausgeschaltet, auch wenn das BS2000-Accounting wieder verfügbar ist. Das UTM-Accounting muss dann mit *account*='Y' erneut eingeschaltet werden.
account='N'	Abrechnungsphase des Accounting ausschalten (OFF).
calc='Y'	Kalkulationsphase des UTM-Accounting einschalten (ON).
calc='N'	Kalkulationsphase des UTM-Accounting ausschalten (OFF).

  Wirkungsdauer / Transaktionssicherung: Typ GIR ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

  Nach dem Start der Anwendung gilt der bei der KDCDEF-Generierung in ACCOUNT ACC= eingestellte Wert.

- den Messmonitor KDCMON ein- oder ausschalten.

  Zum Messmonitor KDCMON und den UTM-Tools zur Auswertung der Messwerte (KDCEVAL) siehe openUTM-Handbuch „Einsatz von UTM-Anwendungen“.

Feldname	Bedeutung
kdcmon='Y'	KDCMON einschalten (ON)
kdcmon='N'	KDCMON ausschalten (OFF)

  Wirkungsdauer / Transaktionssicherung: Typ IR ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

  KDCDIAG ("KDCDIAG - Diagnosehilfen ein- und ausschalten") / KDCAPPL ("KDCAPPL - Eigenschaften und Grenzwerte für den Betrieb ändern")

- die Protokolldateien für die UTM-Anwendung umschalten.

  Die Protokolldateien der Anwendung (SYSOUT und SYSLST bzw. *stderr* und *stdout*) können im laufenden Betrieb umgeschaltet werden. Sie verhindern damit einen Plattenengpass und ermöglichen die Auswertung und Archivierung der Protokolldateien bei laufender Anwendung.

Feldname	Bedeutung
sysprot_switch='Y'	Die Protokolldateien werden umgeschaltet.

  Wirkungsdauer / Transaktionssicherung: Typ GA ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

  KDCAPPL ("KDCAPPL - Eigenschaften und Grenzwerte für den Betrieb ändern")

- Nur auf BS2000-Sysemen: das STXIT-Logging ein- oder ausschalten

Feldname	Bedeutung
stxit_log='Y'	Das Stxit-Logging wird eingeschaltet.
stxit_log='N'	Das Stxit-Logging wird ausgeschaltet.

  Ist das STXIT-Logging eingeschaltet, dann werden bei Eintritt eines STXIT-Ereignisses mehrere K099-Meldungen auf SYSOUT ausgegeben.

  Wirkungsdauer / Transaktionssicherung: Typ IR ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

  KDCDIAG ("KDCDIAG - Diagnosehilfen ein- und ausschalten")

- Debug-Informationen für den Anschluss an die Datenbank ausgeben.

  Sie können angeben, wohin und in welchem Ausmaß Aufrufe an die XA-Schnittstelle protokolliert werden sollen.

Feldname	Bedeutung
xa_debug='Y'	XA-DEBUG wird eingeschaltet (ON). Die Aufrufe an die XA-Schnittstelle werden protokolliert.
xa_debug='A'	Erweitertes XA-DEBUG (ALL). Zusätzlich zu den Aufrufen an die XA-Schnittstelle werden bestimmte Datenbereiche ausgegeben.
xa_debug='N'	XA-DEBUG wird ausgeschaltet (OFF).
xa_debug_out='S'	Ausgabe auf SYSOUT/stderr.
xa_debug_out='F'	Ausgabe in eine Datei.

  Wenn Sie nur das Feld *xa_debug* verwenden, ohne *xa_debug_out* zu versorgen, so wird ggf. der Wert verwendet, den Sie beim Starten der UTM-Anwendung im Startparameter angegeben haben (siehe openUTM-Handbuch „Einsatz von UTM-Anwendungen“). Andernfalls wird auf SYSOUT/stderr protokolliert.

  Wirkungsdauer / Transaktionssicherung: Typ IR ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

  KDCDIAG ("KDCDIAG - Diagnosehilfen ein- und ausschalten")

11.2.9.21 obj_type = KC_MAX_PAR

Es sollen Anwendungsparameter und Maximalwerte der Anwendung geändert werden. Über den Datenbereich müssen Sie die Datenstruktur *kc_max_par_str* legen.

Mögliche Modifikationen

Alle im Folgenden beschriebenen Modifikationen können in einem Aufruf erfolgen.

- Sie können Maximalwerte der Anwendung ändern, die bei der KDCDEF-Generierung in der Anweisung MAX festgelegt wurden. Die Änderungen können sich auf die Performance der Anwendung auswirken (siehe im Abschnitt „Performance-Kontrolle").

  Den folgenden Tabellen können Sie entnehmen, welche Maximalwerte modifiziert werden können und in welchen Feldern der Datenstruktur *kc_max_par_str* Sie die neuen Maximalwerte übergeben müssen.

Feldname	Bedeutung
bretrynr[5]	*Nur auf BS2000-Systemen:* In *bretrynr* geben Sie an, wie oft UTM versuchen soll, eine Nachricht an das Transportsystem (BCAM) zu übergeben, wenn BCAM die Nachricht nicht sofort übernehmen kann. Der Wert von *bretrynr* sollte nicht zu groß gewählt werden, da der Prozess, der die Nachricht an BCAM übergeben will, für die Dauer der Versuche blockiert ist. Bei der Ausgabe von Asynchron-Nachrichten an einen Dialog-Partner mit *ptype* ='APPLI' hat *bretrynr* keine Bedeutung (siehe *bretrynr* im Abschnitt "kc_max_par_str - Maximalwerte der Anwendung (MAX-Parameter)") Minimalwert: '1' Maximalwert: '32767'
cachesize_paging[3]	In *cachesize_paging* geben Sie an, wieviel Prozent des Cache bei Engpass-Situationen auf einmal auf die KDCFILE geschrieben werden sollen, damit der Speicherplatz im Cache für andere Daten verwendet werden kann. UTM lagert bei einem Paging mindestens 8 UTM-Seiten aus, auch wenn der Wert von *cachesize_paging* kleiner ist. Minimalwert: '0', d.h. es werden 8 UTM-Seiten ausgelagert Maximalwert: '100' (%) Die Größe des Cache wird bei der KDCDEF-Generierung in der MAX-Anweisung festgelegt und kann z.B. mit KC_GET_OBJECT für *obj_type*=KC_MAX_PAR ermittelt werden (*cache_size_pages*).

  Wirkungsdauer / Transaktionssicherung: Typ GIR ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

Feldname	Bedeutung
conn_users[10]	Mit *conn_users* können Sie eine Überlastung der Anwendung durch zu viele aktive Benutzer verhindern. Dazu geben Sie in *conn_users* die maximale Anzahl der Benutzer bzw. Clients an, die gleichzeitig bei der UTM-Anwendung angemeldet sein dürfen. Bei Anwendungen, die mit Benutzerkennungen generiert sind, gilt: ○ Wird für *conn_users* eine Zahl angegeben, die größer als die Zahl der generierten Benutzer ist, dann hat *conn_users* keine Wirkung. ○ Benutzerkennungen, die mit Administrationsberechtigung generiert wurden, können sich auch dann noch an die UTM-Anwendung anmelden, wenn die maximale Anzahl der gleichzeitig arbeitenden Benutzerkennungen bereits erreicht wurde. Bei Anwendungen, die ohne Benutzerkennungen generiert sind, gilt: ○ Durch *conn_users* wird die Anzahl der Dialog-Partner beschränkt, die gleichzeitig mit der UTM-Anwendung verbunden sein können. ○ Wird für *conn_users* eine Zahl angegeben, die größer als die Zahl der generierten Dialog-LTERM-Partner ist, dann hat *conn_users* keine Wirkung. Dialog-LTERM-Partner sind alle LTERM-Partner, die mit *usage_type*='D' eingetragen sind, LTERM-Partner der LTERM-Pools und die LTERM-Partner, die UTM intern für Multiplexanschlüsse erzeugt. Soll die Anzahl der gleichzeitig aktiven Benutzer nicht beschränkt werden bzw. eine Beschränkung aufgehoben werden, dann geben Sie *conn_users*='0' an. Minimalwert: '0' (d.h. keine Beschränkung) Maximalwert: '500000' Auf Unix-, Linux- und Windows-Systemen darf der Maximalwert nicht größer sein als der im Generierungsparameter MAX ... CONN-USERS generierte Wert.

Wirkungsdauer / Transaktionssicherung: Typ IR ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

- Sie können ein neues Ziel für die Ergebnisse der Administrationskommandos von KDCADM definieren, die über Asynchron-TACs von KDCADM aufgerufen wurden.

Feldname	Bedeutung
destadm[8]	In *destadm* geben Sie den neuen Empfänger für die Ergebnisse von KDCADM-Administrationsaufrufen an, die asynchron verarbeitet wurden (Asynchron-Transaktionscodes von KDCADM). Der alte Wert von *destadm* wird damit überschrieben. Für *destadm* können Sie Folgendes angeben: ○ den Namen eines LTERM-Partners oder ○ einen Asynchron-Transaktionscode oder ○ eine TAC-Queue Wenn Sie für *destadm* Leerzeichen angeben, dann ist kein Empfänger mehr definiert. Die Ergebnisse der Asynchron-Transaktionscodes von KDCADM gehen dann verloren.

  Wirkungsdauer / Transaktionssicherung: Typ GPD ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

- Sie können die Anzahl der Fehlversuche ändern, die UTM bei der Anmeldung erlaubt, bevor UTM stillen Alarm auslöst.

Feldname	Bedeutung
signon_fail	In *signon_fail* geben Sie die Anzahl unmittelbar aufeinanderfolgender erfolgloser Anmeldeversuche (Sicherheitsverletzungen) von einem Client aus an, nach der ein "stiller Alarm" (K094-Meldung) ausgelöst wird. Minimalwert: '1' Maximalwert: '100'

  Wirkungsdauer / Transaktionssicherung: Typ GIR ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

- Sie können die Datenlieferung an openSM2 ein- bzw. ausschalten.

Feldname	Bedeutung
sm2='Y'	UTM soll zur Überwachung der Performance Daten an openSM2 liefern. Die Datenlieferung kann nur eingeschaltet werden, wenn sie nicht bei der KDCDEF-Generierung generell ausgeschlossen wurde (MAX-Anweisung Operand SM2).
sm2='N'	Die Datenlieferung an openSM2 soll ausgeschaltet werden.

  Wirkungsdauer / Transaktionssicherung: Typ GIR ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

Einige der Modifikationen sind auch mit dem Administrationskommando KDCAPPL ("KDCAPPL - Eigenschaften und Grenzwerte für den Betrieb ändern") durchführbar.

11.2.9.22 obj_type=KC_TASKS_PAR

Es können die Werte geändert werden, die sich auf die Prozessanzahl der Anwendung beziehen, d.h. Gesamtzahl der Prozesse, Maximalzahl der Prozesse für die Bearbeitung von Asynchron-Aufträgen und Teilprogrammen mit blockierenden Aufrufen sowie die Anzahl der Prozesse, die für UTM-interne Aufgaben und Dialog-Aufträge, die keiner TAC-Klasse zugeordnet sind, freigehalten werden.
Über den Datenbereich müssen Sie die Datenstruktur *kc_tasks_par_str* legen.

Mögliche Modifikationen

Alle im Folgenden beschriebenen Modifikationen können in einem Aufruf erfolgen.

Feldname	Bedeutung
mod_max_tasks[3]	Gesamtzahl der laufenden Prozesse ändern: In diesem Feld geben Sie die maximale Anzahl der Prozesse an, die für die Anwendung laufen. *mod_max_tasks* ist ein Sollwert für die aktuelle Prozessanzahl. Die Anzahl der tatsächlich aktiven Prozesse, die aktuell Aufträge der Anwendung bearbeiten, ist im Feld *curr_tasks* abgelegt (siehe *kc_tasks_par_str* im Abschnitt "kc_tasks_par_str - Anzahl der Prozesse"). Diese kann beim Starten oder Beenden eines Prozesses kurzfristig anders sein als *mod_max_tasks*. Maximalwert: der bei der KDCDEF-Generierung in MAX festgelegte Maximalwert (*tasks*) Minimalwert: '1'
mod_max_asyntasks[3]	Maximalzahl der Prozesse ändern, die gleichzeitig Asynchron-Aufträge bearbeiten dürfen. In *mod_max_asyntasks* geben Sie die maximale Anzahl der Prozesse an, die gleichzeitig für Asynchron-Verarbeitung verwendet werden dürfen. Die hier angegebene Anzahl entspricht einem oberen Grenzwert. Die tatsächliche Maximalzahl der Prozesses, die gleichzeitig für Asynchron-Verarbeitung verwendet werden dürfen (siehe *kc_tasks_par_str* im Abschnitt "kc_tasks_par_str - Anzahl der Prozesse", Parameter *curr_max_asyntasks*), kann kleiner sein als der in *mod_max_asyntasks* angegebene Wert, da die tatsächliche Anzahl durch die Anzahl der aktuell laufenden Prozesse der Anwendung (*curr_tasks*) begrenzt ist. Minimalwert: '0' Maximalwert: die bei der KDCDEF-Generierung in MAX definierte Maximalzahl (*asyntasks)*

Feldname	Bedeutung
mod_max_tasks_in_pgwt[3]	Maximalzahl der Prozesse ändern, in denen gleichzeitig Teilprogramme mit blockierenden Aufrufen ablaufen dürfen. In *mod_max_tasks_in_pgwt* geben Sie die gewünschte maximale Anzahl der Prozesse an. Die hier angegebene Anzahl entspricht einem oberen Grenzwert. Die tatsächliche Maximalzahl der Prozesses, die gleichzeitig Teilprogramme mit blockierenden Aufrufen bearbeiten (siehe *kc_tasks_par_str* im Abschnitt "kc_tasks_par_str - Anzahl der Prozesse", Parameter *curr_max_tasks_in_pgwt*), kann kleiner sein als der in *mod_max_tasks_in_pgwt* angegebene Wert, da die tatsächliche Anzahl mindestens 1 kleiner sein muss als die Anzahl der aktuell laufenden Prozesse der Anwendung (*curr_tasks*). *mod_max_tasks_in_pgwt*='0' wird abgewiesen, wenn in der Anwendung Transaktionscodes oder TAC-Klassen mit *pgwt*='Y' definiert sind. Minimalwert: '0' Maximalwert: die bei der KDCDEF-Generierung in MAX definierte Maximalzahl (*tasks_in_pgwt)*
mod_free_dial_tasks[3]	kann nur geändert werden, wenn bei der KDCDEF-Generierung eine TAC-PRIORITIES-Anweisung abgesetzt wurde. In *mod_free_dial_tasks* geben Sie an, wieviele Prozesse der Anwendung für die Bearbeitung von UTM-internen Aufgaben und für Aufträge, die keiner Dialog-TAC-Klasse zugeordnet sind, freigehalten werden sollen. Die maximale Anzahl der Prozesse, die gleichzeitig Asynchron-Aufträge bearbeiten dürfen, wird durch *mod_free_dial_tasks* nicht beschränkt Ist nach einer Änderung der Prozesszahlen *mod_free_dial_tasks* >= *mod_max_tasks*, dann darf trotzdem ein Prozess der Anwendung Aufträge an Dialog-TAC-Klassen bearbeiten. Minimalwert: '0' Maximalwert: Wert in *tasks* -1

Wirkungsdauer / Transaktionssicherung: Typ A ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

KDCAPPL ("KDCAPPL - Eigenschaften und Grenzwerte für den Betrieb ändern")

11.2.9.23 obj_type = KC_TIMER_PAR

Es sollen die Einstellungen von Timern der Anwendung geändert werden. Über den Datenbereich müssen Sie die Datenstruktur *kc_timer_par_str* legen.

Mögliche Modifikationen

Der folgenden Tabelle können Sie entnehmen, welche Timer modifiziert werden können. In einem Aufruf können Sie beliebig viele dieser Timer ändern.

Feldname	Bedeutung
conrtime_min[5]	Hier geben Sie die Zeit in Minuten an, nach der UTM versuchen soll, eine unterbrochene Verbindung zu einem Drucker oder einer TS-Anwendung wieder aufzubauen. Voraussetzung ist, dass die Verbindung von UTM zuvor automatisch aufgebaut wurde (*kc_pterm_str*.*auto_connect*='Y' oder *kc_lterm_str*.*plev* > 0). Bei *conrtime_min*='0' unternimmt UTM keinen Versuch, eine unterbrochene Verbindung wieder aufzubauen. Maximalwert: '32767' Minimalwert: '0'
pgwttime_sec[5]	Zeit in Sekunden, die ein Teilprogramm nach einem blockierenden Funktionsaufruf (z.B. PGWT) maximal auf das Eintreffen von Nachrichten warten darf. Während dieser Wartezeit bleibt ein Prozess exklusiv durch dieses Teilprogramm belegt. Maximalwert: '32767' Minimalwert: '60'
reswait_ta_sec[5]	Zeit in Sekunden, die ein Teilprogramm maximal auf ein von einer anderen Transaktion gesperrtes Betriebsmittel warten soll. *reswait_ta_sec*='0' bedeutet, dass nicht gewartet wird. Ein Teilprogrammlauf, der auf gesperrte Betriebsmittel zugreifen will, erhält sofort einen entsprechenden Returncode. Maximalwert: '32767' Minimalwert: '0'

Feldname	Bedeutung
reswait_pr_sec[5]	Zeit in Sekunden, die maximal auf ein von einem anderen Prozess gesperrtes Betriebsmittel gewartet werden soll. Wird diese Zeit überschritten, dann beendet sich die Anwendung mit einer Fehlermeldung. Es ist zu beachten, dass der Wert von *reswait_pr_sec* mindestens so groß sein muss wie die längste Bearbeitungszeit (Realzeit) für die folgenden Fälle: • Bei TS-Anwendungen, die keine SOCKET-Anwendungen sind (Clients mit PTYPE=APPLI), sind die Betriebsmittel für die Dauer eines Verarbeitungsschrittes inklusive VORGANG-Exit bei Vorgangsbeginn und/oder Vorgangsende gesperrt. • Bei Vorgangsende sind die Betriebsmittel gesperrt, solange das VORGANG-Exit-Programm läuft. Minimalwert: '300', Maximalwert: '32767' Geben Sie einen Wert < 300 an, dann wird der Aufruf abgewiesen.
termwait_in_ta_sec[5]	Zeit in Sekunden, die bei einer Mehrschritt-Transaktion (d.h. im Programm PEND KP) maximal zwischen einer Ausgabe an einen Dialog-Partner und der nachfolgenden Dialog-Antwort verstreichen darf. Wird die Zeit *termwait_in_ta_sec* überschritten, dann wird die Transaktion zurückgesetzt. Die von der Transaktion reservierten Betriebsmittel werden freigegeben. Die Verbindung zum Partner wird abgebaut. Maximalwert: '32767' Minimalwert: '60'
logackwait_sec[5]	*Nur auf BS2000-Systemen:* Zeit in Sekunden, die UTM maximal auf eine logische Abdruckquittung vom Drucker bzw. auf eine Transportquittung für eine Asynchron-Nachricht an eine andere Anwendung (erzeugt mit dem KDCS-Aufruf FPUT) warten soll. Trifft nach dieser Zeit die Quittung nicht ein, z.B. wegen Papierende bei Druckern, baut UTM die logische Verbindung zu dem Gerät ab. Minimalwert: '10' Maximalwert: '32767'

Feldname	Bedeutung
Folgende Timer haben nur bei UTM-Anwendungen mit verteilter Verarbeitung über LU6.1 bzw. OSI TP eine Bedeutung.	
conctime1_sec[5]	Zeit in Sekunden zur Überwachung des Aufbaus einer Session (LU6.1) bzw. Association (OSI TP). Wird die Session bzw. Association innerhalb der angegebenen Zeit nicht aufgebaut, dann baut UTM die Transportverbindung zur Partner-Anwendung ab. *conctime1_sec*='0' bedeutet: • bei LU6.1-Verbindungen, dass der Aufbau einer Session nicht überwacht wird (es wird beliebig lange gewartet), • bei OSI TP-Verbindungen, dass maximal 60 Sekunden auf den Aufbau einer Association gewartet wird. Minimalwert: '0', Maximalwert: '32767'
conctime2_sec[5]	Zeit in Sekunden, die beim Übertragen einer Asynchron-Nachricht maximal auf eine Quittung vom Empfänger gewartet wird. Nach Ablauf der Zeit *conctime2_sec* baut UTM die Transportverbindung ab. Der Asynchron-Auftrag geht nicht verloren, er steht weiterhin in der lokalen Message Queue. *conctime2_sec*='0' bedeutet, dass keine Überwachung durchgeführt wird. Minimalwert: '0', Maximalwert: '32767'
ptctime_sec[5]	Der Timer hat nur eine Bedeutung für die verteilte Verarbeitung über LU6.1-Verbindungen. *ptctime_sec* legt die Zeit in Sekunden fest, die ein lokaler Auftragnehmer-Vorgang maximal im Zustand PTC (prepare to commit, Transaktionsstatus P) auf eine Quittung vom Auftraggeber-Vorgang wartet. Nach Ablauf der Zeit wird die Verbindung zum Auftraggeber abgebaut, die Transaktion im Auftragnehmer-Vorgang zurückgesetzt und der Vorgang beendet. Dadurch kann es evtl. zu einem Mismatch kommen. Wurde für die Anwendung bereits KDCSHUT WARN oder GRACE gegeben und der Wert von *ptc_time_sec* ist ungleich 0, so wird die Wartezeit unabhängig von *ptc_time_sec* so gewählt, dass die Transaktion zurückgesetzt wird, bevor die Anwendung beendet wird, um eine abnormale Anwendungsbeendigung mit ENDPET möglichst zu vermeiden. *ptctime_sec*='0' bedeutet, dass beliebig lange auf eine Quittung gewartet wird. Minimalwert: '0' Maximalwert: '32767'

Weitere Informationen siehe Abschnitt "kc_timer_par_str - Timer-Einstellungen".

Wirkungsdauer / Transaktionssicherung: Typ GIR ("KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern")

Die Änderungen sind für bereits laufende Timer nicht wirksam, sie wirken erst für Timer, die nach der Änderung aufgezogen werden.

Einige der Modifikationen sind auch mit dem Administrationskommando KDCAPPL ("KDCAPPL - Eigenschaften und Grenzwerte für den Betrieb ändern") durchführbar.

11.2.9.24 Returncodes

Neben den im Abschnitt „Returncodes" aufgelisteten Returncodes können zusätzlich folgende Returncodes auftreten. Einige dieser Returncodes können unabhängig vom angegebenen Objekttyp auftreten, andere treten nur für bestimmte Objekttypen auf.

Typunabhängige Returncodes:

Maincode = KC_MC_DATA_INVALID

In der Datenstruktur im Datenbereich fehlen Angaben bzw. ein Feld enthält einen ungültigen Wert.

Subcodes:

KC_SC_DATA_MISSING

In der Datenstruktur fehlen Angaben. Mögliche Ursachen:

- Es wurde kein Feld angegeben, das modifiziert werden soll.
- Für die angeforderte Änderung müssen mehrere Felder zusammen angegeben werden und es fehlt eine dieser Angaben, z.B. *obj_type*=KC_TPOOL: *state* und *state_number*.

KC_SC_INVALID_MOD

In der Datenstruktur wurde ein Feld, das modifiziert werden kann, mit einem ungültigen Wert versorgt.

KC_SC_NOT_NULL

In der Datenstruktur wurde ein Feld, das nicht modifiziert werden kann, nicht mit binär null versorgt.

Maincode = KC_MC_REJECTED_CURR

Der Aufruf kann zur Zeit nicht bearbeitet werden.

Subcode:

KC_SC_INVDEF_RUNNING

Es läuft z. Zt. ein inverser KDCDEF, während des Laufs dürfen keine Konfigurationsdaten verändert werden.

Maincode = KC_MC_NOT_EXISTENT

Es existiert kein Objekt des in *obj_type* angegebenen Objekttyps mit dem im Identifikationsbereich übergebenen Namen bzw. Namenstripel.

Subcode:

KC_SC_NO_INFO

Maincode = KC_MC_DELETED

Das angegebene Objekt ist gelöscht. Seine Eigenschaften können nicht modifiziert werden.

Subcode:

KC_SC_NO_INFO

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen.

Subcode:

KC_SC_NOT_GEN

Es existiert kein explizit erzeugtes Objekt des in *obj_type* angegebenen Objekttyps. Es können aber implizit erzeugte Objekte existieren, z.B. Benutzerkennungen für Clients mit *ptype*='APPLI'.

KC_SC_JCTL_RT_CODE_NOT_OK

Nur bei UTM-Cluster-Anwendungen:
Beim globalen Modifizieren eines Objekts ist ein UTM-interner Fehler aufgetreten. Bitte wenden Sie sich an die Systembetreuung.

KC_SC_NO_CLUSTER_APPLI

Diese Aktion ist nur bei einer UTM-Cluster-Anwendung möglich.

KC_SC_NO_GLOB_CHANG_POSSIBLE

Keine globalen Administrationsänderungen möglich, da die Generierungen der Knoten-Anwendungen zur Zeit nicht konsistent sind.

KC_SC_NOT_ALLOWED_IN_CLUSTER

Die Administrationsaktion ist in einer UTM-Cluster-Anwendung nicht erlaubt.

Maincode = KC_MC_RECBUF_FULL

Der Puffer mit Recovery-Information ist voll (siehe KDCDEF-Steueranweisung MAX, Operand RECBUF).

Subcode:

KC_SC_NO_INFO

Returncodes bei obj_type = KC_CLUSTER_NODE:

Maincode = KC_MC_REJECTED Der Aufruf wurde von UTM abgewiesen. **Subcode:**
KC_SC_CCFG_NO_CLUSTER_APPLI Bei der angegebenen Anwendung handelt es sich nicht um eine UTM-Cluster-Anwendung
KC_SC_CCFG_FILE_NOT_OPEN UTM-interner Fehler. Bitte wenden Sie sich an die Systembetreuung.
KC_SC_CCFG_RT_CODE_NOT_OK Modifikation wurde nicht durchgeführt. Mögliche Ursache z.B. Timer abgelaufen.
KC_SC_CCFG_FILE_LOCK_ERROR Cluster-Konfigurationsdatei ist gesperrt.
KC_SC_CCFG_FILE_READ_ERROR Fehler beim Lesen der Cluster-Konfigurationsdatei.
KC_SC_CCFG_FILE_WRITE_ERROR Fehler beim Schreiben der Cluster-Konfigurationsdatei.
KC_SC_CCFG_INVALID_BUFFER_LTH UTM-interner Fehler. Bitte wenden Sie sich an die Systembetreuung.
KC_SC_CCFG_INVALID_NODE_INDEX UTM-interner Fehler. Bitte wenden Sie sich an die Systembetreuung.
KC_SC_CCFG_INVALID_NODE_STATE Ungültiger Status der Knoten-Anwendung. Hinweis: Für eine laufende Knoten-Anwendung darf nichts geändert werden.
KC_SC_CCFG_INVAL_FILEBASE_NAME Basisname der UTM-Cluster-Anwendung ungültig.
KC_SC_CCFG_INVALID_HOSTNAME Der Rechnername ist ungültig.

Returncodes bei obj_type = KC_DB_INFO:

Maincode = KC_MC_REJECTED Der Aufruf wurde von UTM abgewiesen. **Subcode:**
KC_SC_NOT_GEN Für die Anwendung ist keine Datenbank generiert.
KC_SC_INVALID_TYPE Bei der im Identifikationsbereich ausgewählten Datenbank handelt es sich nicht um eine XA-Datenbank.
KC_SC_NO_INFO Interner Fehler in UTM beim Codieren des neuen Passwortes.

Maincode = KC_MC_NOT_EXISTENT Das im Identifikationsbereich angegebene Objekt existiert nicht. **Subcode:**
KC_SC_NO_INFO

Returncodes bei obj_type = KC_KSET:

Maincode = KC_MC_REJECTED Der Aufruf wurde von UTM abgewiesen. **Subcode:**
KC_SC_NOT_ALLOWED Das Ändern des Keysets KDCAPLKS bzw. MASTER ist nicht erlaubt.

Returncodes bei obj_type = KC_LOAD_MODULE (Programmaustausch):

Maincode = KC_MC_REJECTED_CURR Der Aufruf kann zur Zeit nicht bearbeitet werden. **Subcode:**
KC_SC_CHANGE_RUNNING Es läuft gerade ein Programmaustausch.

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen.

Subcodes:

KC_SC_NOT_CHANGEABLE

Das im Identifikationsbereich angegebene Lademodul/Shared Object/DLL ist nicht austauschbar. Mögliche Gründe sind z.B.:

- Das Lademodul hat den Lademodus STATIC.
- Das Lademodul enthält TCB-Entries.

KC_SC_SAME_VERSION

load_mode !='U' (nicht STARTUP):
In *version* wurde die aktuell geladene Version des Lademoduls angegeben.

KC_SC_LMOD_NOT_EXISTENT (nur auf BS2000-Systemen)

In der Bibliothek konnte kein Modul mit der angegebenen Version gefunden werden.

KC_SC_INVALID_VALUE (nur auf BS2000-Systemen)

Der Lademodul ist mit LOAD-MODE=POOL, (POOL,STARTUP) oder (POOL,ONCALL) sowie mit Version *HIGHEST-EXISTING generiert, aber bei *version* wurde ein Wert ungleich *HIGHEST-EXISTING angegeben.

Returncodes bei obj_type = KC_LPAP:

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen.

Subcodes:

KC_SC_CONNECTED

state='N': Es besteht eine Verbindung zu der Partner-Anwendung. Die Partner-Anwendung kann deshalb nicht gesperrt werden. Vor dem Sperren der Partner-Anwendung müssen alle Verbindungen zu ihr abgebaut werden.

KC_SC_NOT_ALLOWED

Mögliche Ursachen:

- Sie haben versucht mit *connect_mode*='Y' eine Verbindung zu einer gesperrten Partner-Anwendung (*state*='N') aufzubauen, oder
- Sie haben *state*='N' zusammen mit *connect_mode*='Y' gesetzt, oder
- Sie haben *connect_mode* und *quiet_connect* zusammen angegeben, oder
- der in *bcam_trace* angegebene Wert ist nicht zulässig.

KC_SC_NOT_EXISTENT

Das angegebene Objekt existiert nicht.

Returncodes bei obj_type = KC_LSES:

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen.

Subcodes:

KC_SC_NOT_ALLOWED

Mögliche Ursachen:

- Die Kombination der angegebenen Modifikationen ist nicht erlaubt, d.h. es wurde sowohl *connect_mode* als auch *quiet_connect* gesetzt.
- Eine Verbindung zur Partner-Anwendung existiert nicht und kann nicht aufgebaut werden, weil der LPAP-Partner der Partner-Anwendung gesperrt ist. Der LPAP-Partner muss zuerst in einer eigenen Transaktion entsperrt werden.

KC_SC_INVALID_CON

Die durch (*con*, *pronam*, *bcamappl*) angegebene Verbindung ist ungültig. Sie existiert nicht oder ist für eine andere Partner-Anwendung (LPAP-Partner) vorgesehen.

KC_SC_CONNECTED

In (*con*, *pronam*, *bcamappl*) wurde eine Verbindung angegeben, die aufgebaut werden soll. Die Session besteht jedoch bereits über eine andere Verbindung.

Maincode = KC_MC_NOT_EXISTENT

Das angegebene Objekt existiert nicht.

Subcode:

KC_SC_NO_INFO

Es wurde keine LU6.1-Verbindung erzeugt bzw. generiert.

Returncodes bei obj_type = KC_LTAC:

Zu KC_LTAC gibt es keine Typ-spezifischen Returncodes.

Returncodes bei obj_type = KC_LTERM:

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen.

Subcodes:

KC_SC_POOL_LTERM

Der im Identifikationsbereich angegebene LTERM-Partner gehört zu einem LTERM-Pool. Die angeforderte Modifikation ist für diesen LTERM-Partner nicht zulässig.

KC_SC_NO_PTERM

Es wurde *connect_mode*='Y' gesetzt:
UTM kann keine Verbindung aufbauen, da dem LTERM-Partner zur Zeit kein Client/Drucker zugeordnet ist oder der zugehörige Client/Drucker gesperrt ist.

KC_SC_NOT_ALLOWED

Mögliche Ursachen:

- Es wurde versucht, für einen LTERM-Partner mit *usage_type*='O' ein Startformat zu definieren.
- Es wurde *format_attr*='E' (#Format) angegeben, es ist aber kein Anmelde-Vorgang definiert.
- In *bcam_trace* wurde ein unzulässiger Wert angegeben.
- Austausch zweier Master-LTERMs wurde abgewiesen, weil eines der LTERMs kein Master-LTERM ist, oder für beide der gleiche Master angegeben wurde. Der Austausch zweier Master-LTERMs ist in einer UTM-Cluster-Anwendung nicht erlaubt.

KC_SC_NO_FORMAT_ALLOWED

Angaben in *format_name* und *format_attr* (Ändern des Startformats) sind nicht erlaubt, da für die Anwendung kein Formatierungssystem generiert wurde.

KC_SC_INVALID_ALIAS

Das Primary-LTERM ist selbst ein Alias-LTERM.

KC_SC_INVALID_ALIAS_CTERM

Das Primary-LTERM ist ein CTERM.

KC_SC_INVALID_ALIAS_BUNDLE

Das Primary-LTERM ist ein Slave-LTERM in einem LTERM-Bündel.

KC_SC_ALIAS_STATE_ILL

Das Primary-LTERM ist mit RESTART=NO oder QAMSG=NO genereriert.

Returncodes bei obj_type = KC_MUX (BS2000-Systeme)

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen.

Subcodes:

KC_SC_CONNECTED

state='N': Es besteht eine Verbindung zu dem Multiplexanschluss. Er kann deshalb nicht gesperrt werden.
connect_mode='Y': Es besteht bereits eine Verbindung zu dem Multiplexanschluss.

KC_SC_NOT_ALLOWED

Sie haben versucht eine Verbindung zu einem gesperrten Multiplexanschluss aufzubauen oder der angegebene Wert in *bcam_trace* ist nicht erlaubt.

Returncodes bei obj_type = KC_OSI_CON:

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen.

Subcode:

KC_SC_CONNECTED

Es besteht eine Verbindung zu der Partner-Anwendung. Auf eine Ersatzverbindung kann nur umgeschaltet werden, wenn zu der Partner-Anwendung keine aktive Association besteht.

Returncodes bei obj_type = KC_OSI_LPAP:

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen.

Subcodes:

KC_SC_CONNECTED

Angabe *state*='N': Es besteht eine Verbindung zu der Partner-Anwendung. Der OSI-LPAP-Partner der Partner-Anwendung kann deshalb nicht gesperrt werden. Vor dem Sperren müssen alle Verbindungen zur Partner-Anwendung abgebaut werden.

KC_SC_NOT_ALLOWED

Mögliche Ursachen:

- Sie haben versucht eine Verbindung (*connect_number* > 0) zu einer gesperrten Partner-Anwendung (OSI-LPAP-Partner) aufzubauen oder zu einer Partner-Anwendung, für die keine Verbindung auf aktiv gesetzt ist (siehe *kc_osi_con_str* Feld *active*)
- Sie haben *state*='N' zusammen mit *connect_number* gesetzt, oder
- Sie haben *state*='N' zusammen mit *quiet_connect* gesetzt, oder
- Sie haben *quiet_connect* zusammen mit *connect_number* gesetzt.

Returncodes bei obj_type = KC_PTERM:

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen.

Subcodes:

KC_SC_NOT_ALLOWED

Mögliche Ursachen:

- Es wurde versucht, eine Verbindung zu einem gesperrten Client/Drucker aufzubauen.
- *connect_mode*='R' ist für den im Identifikationsbereich angegebenen Client nicht erlaubt.
- Die Felder *lterm* und *connect_mode* wurden zusammen angegeben.
- *state*='N' und *auto_connect*='Y' wurden zusammen angegeben.

KC_SC_POOL_PTERM

Die angeforderte Modifikation ist für Clients, die sich über einen LTERM-Pool anschließen, nicht erlaubt.

KC_SC_UPIC_PTERM

Die angeforderte Modifikation ist für Clients mit *ptype*='UPIC-R' bzw. 'UPIC-L' (auf Unix-, Linux- und Windows-Systemen) nicht erlaubt.

KC_SC_TTY_PTERM (nur auf Unix-, Linux- und Windows-Systemen)

Die angeforderte Modifikation ist für ein Terminal (*ptype*='TTY') nicht erlaubt.

KC_SC_MUX_DIS_PENDING (nur auf BS2000-Systemen)

Der angegebene Client ist über einen Multiplexanschluss mit der Anwendung verbunden und die Session ist im Zustand DISCONNECT PENDING.
Es wurde entweder versucht die Session auf- oder abzubauen (*connect_mode*='Y' oder 'N') oder die Session bei noch laufendem Timer explizit freizugeben (*connect_mode*='R').

KC_SC_LTERM_NOT_EXISTENT

Die Zuordnung Client/Drucker zu LTERM-Partner kann nicht geändert werden, da der in *lterm* angegebene LTERM-Partner nicht existiert.

KC_SC_LTERM_DEL

Die Zuordnung Client/Drucker zu LTERM-Partner kann nicht geändert werden, da der in *lterm* angegebene LTERM-Partner gelöscht wurde.

KC_SC_LTERM_NOT_ALLOWED

Die Zuordnung Client/Drucker zu LTERM-Partner kann nicht geändert werden.
Mögliche Ursachen:

- Der in *lterm* angegebene LTERM-Partner gehört zu einem LTERM-Pool.
- Der angegebene LTERM-Partner wurde für den Anschluss eines Client mit *ptype*='UPIC-...' konfiguriert, er kann keinem anderen Client zugeordnet werden.
- In *lterm* wurde KDCMSGLT angegeben. KDCMSGLT wird von UTM intern für den Event-Service MSGTAC erzeugt. Er kann keinem Client/Drucker zugeordnet werden.

KC_SC_CONNECTED

Die Zuordnung Client/Drucker zu LTERM-Partner kann nicht geändert werden.
Mögliche Ursachen:

- Der Client/Drucker, der dem LTERM-Partner zugeordnet werden soll, ist z.Zt. mit der Anwendung verbunden.
- Dem LTERM-Partner ist zur Zeit ein Client zugeordnet, der mit der Anwendung verbunden ist. Die alte Zuordnung des LTERM-Partners kann nicht aufgelöst werden, da einer der beiden Clients als Dialog-Partner eingetragen ist (*usage_type*='D').

KC_SC_OUT_PTERM_DIAL_LTERM

Im Identifikationsbereich wurde der Name eines Ausgabemediums (*usage_type*='O') angegeben, der in *lterm* angegebene LTERM-Partner ist jedoch als Dialog-Partner konfiguriert.
Ein Ausgabemedium kann keinem Dialog-LTERM-Partner zugeordnet werden.

KC_SC_DIAL_PTERM_TO_BUNDLE

Die neue Zuordnung Client/Drucker zu LTERM-Partner kann nicht hergestellt werden.
Im Identifikationsbereich wurde der Name eines Dialog-Partners (*usage_type*='D') übergeben, der in *lterm* angegebene LTERM-Partner gehört aber zu einem Druckerbündel.

KC_SC_PTYPE_APPLI

Die neue Zuordnung Client zu LTERM-Partner kann nicht hergestellt werden.
Im Identifikationsbereich wurde der Name eines Client mit *ptype*='APPLI' oder 'SOCKET' angegeben. Der in *lterm* angegebene LTERM-Partner passt nicht zu diesem Client, weil für den LTERM-Partner keine Benutzerkennung generiert wurde.

KC_SC_PTERM_WITHOUT_CID

Die neue Zuordnung Drucker zu LTERM-Partner kann nicht hergestellt werden.
Der angegebene LTERM-Partner ist einem Druckersteuer-LTERM zugeordnet, aber für den angegebenen Drucker ist keine Drucker-Id (CID) definiert.

KC_SC_CID_AMBIGUOUS

Die neue Zuordnung Drucker zu LTERM-Partner kann nicht hergestellt werden.
Der angegebene LTERM-Partner ist einem Druckersteuer-LTERM zugeordnet, für den angegebenen Drucker ist jedoch eine Drucker-Id(CID) definiert, die im Bereich des Druckersteuer-LTERMs nicht eindeutig ist.

KC_SC_NO_LTERM

connect_mode='Y' nicht erlaubt: Dem angegebenen Client/Drucker ist kein LTERM-Partner zugeordnet. Deshalb kann keine Verbindung aufgebaut werden.

KC_SC_INVALID_PROTOCOL_USAGE

PTYPE und Protokoll sind nicht miteinander vereinbar.

KC_SC_BUNDLE_NOT_ALLOWED

Die neue Zuordnung Client zu LTERM-Partner kann nicht hergestellt werden, weil der LTERM-Partner zu einem LTERM-Bündel gehört.

KC_SC_GROUP_NOT_ALLOWED

Die neue Zuordnung Client zu LTERM-Partner kann nicht hergestellt werden, weil der LTERM-Partner zu einer LTERM-Gruppe gehört.

KC_SC_NOT_ALLOWED_IN_CLUSTER

Diese Funktion ist bei einer UTM-Cluster-Anwendung nicht erlaubt, z.B. KDCSWTCH oder Austausch zweier Bundle-Master.

Returncodes bei obj_type = KC_TAC:

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen.

Subcodes:

KC_SC_NOT_ALLOWED

Mögliche Ursachen:

- Es wurde versucht *state* zu modifizieren und gleichzeitig Statistikwerte zurückzusetzen.
- Möglicherweise wurde versucht die Parameter *lock_code* und *access_list* zu verändern. Das Veränderung von *access_list* ist nicht erlaubt, wenn *lock_code* generiert ist.
- Das Verändern von *access_list* ist bei den TACs KDCBADTC, KDCMSGTC und KDCSGNTC nicht erlaubt.
- Es wurde versucht KDCTAC zu sperren.
- Ein mit der Eigenschaft NEXT generierter TAC sollte mit *state*='N' gesperrt werden. Dies ist nicht zulässig. Die Sperre hätte keine Wirkung.
- Bei einem TAC, der nicht vom Typ 'Q' ist, wurde versucht 'q_read_acl' oder 'q_write_acl' zu verändern.
- Es wurde versucht, *dead_letter_q* = 'Y' zu setzen für einen Dialog- oder Asynchron-TAC mit CALL=NEXT, oder für einen TAC KDCDLETQ oder KDCMSGTC.

KC_SC_INVALID_READ_ACL

Das in *q_read_acl* angegebene Keyset existiert nicht.

KC_SC_INVALID_WRITE_ACL

Das in *q_read_acl* angegebene Keyset existiert nicht.

KC_SC_INVALID_ACL

Das in *access_list* angegebene Keyset existiert nicht.

KC_SC_READ_ACL_DEL

Das Keyset wurde gelöscht.

KC_SC_WRITE_ACL_DEL

Das Keyset wurde gelöscht.

Returncodes bei obj_type = KC_TACCLASS:

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen.

Subcodes:

KC_SC_NOT_ALLOWED

- In *tasks* oder *tasks_free* wurde eine ungültige Anzahl an Prozessen angegeben.
- Es wurde sowohl *tasks* als auch *tasks_free* angegeben.

KC_SC_NOT_CHANGEABLE

tasks und *tasks_free* dürfen nicht geändert werden, weil die Anwendung mit Prioritätensteuerung (TAC-PRIORITIES) generiert wurde.

Returncodes bei obj_type = KC_TPOOL:

Zu KC_TPOOL gibt es keine Typ-spezifischen Returncodes.

Returncodes bei obj_type = KC_USER:

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen.

Subcodes:

KC_SC_TOO_SIMPLE

Die angeforderte Passwort-Änderung wird nicht durchgeführt, da das neue Passwort nicht der Komplexitätsstufe (*protect_pw_compl*) entspricht, die für die Benutzerkennung definiert wurde.

KC_SC_OLD_PW

Die angeforderte Passwort-Änderung wird nicht durchgeführt, da in *password16* das alte Passwort angegeben wurde und für die Benutzerkennung eine begrenzte Gültigkeitsdauer des Passworts definiert ist (*protect_pw_time* !='0'). Bei der Benutzerkennung darf nicht das alte Passwort als neues Passwort angegeben werden.

KC_SC_NOT_ALLOWED

Die angeforderte Modifikation wurde nicht durchgeführt. Mögliche Ursachen:

- *state*='N': Sie haben versucht, eine Benutzerkennung mit Administrationsberechtigung (permit='A' oder 'B') zu sperren.
- Sie haben versucht, eine Benutzerkennung zu modifizieren, die einem Client mit *ptype*='APPLI', 'SOCKET' oder 'UPIC-...'zugeordnet ist.
- Sie haben versucht, die Benutzerkennung KDCMSGUS zu modifizieren, die UTM intern für den Event-Exit MSGTAC erzeugt hat.

- Sie haben *format_attr*='E' (#Format) angegeben, es ist aber kein Anmelde-Vorgang definiert.
- Ein- oder Ausschalten des BCAM-Trace ist nur erlaubt, wenn der BTRACE-Mode auf SELECT-Mode gestellt ist.

KC_SC_NO_FORMAT_ALLOWED

Angaben in *format_name* und *format_attr* (Ändern des Startformats) sind nicht erlaubt, da für die Anwendung kein Formatierungssystem generiert wurde.

KC_SC_INVALID_READ_ACL

Das in *q_read_acl* angegebene Keyset existiert nicht.

KC_SC_INVALID_WRITE_ACL

Das in *q_write_acl* angegebene Keyset existiert nicht.

KC_SC_READ_ACL_DEL

Das referenzierte Keyset wurde gelöscht.

KC_SC_WRITE_ACL_DEL

Das referenzierte Keyset wurde gelöscht.

KC_SC_KSET_DEL

Der angegebene Keyset wurde gelöscht.

KC_SC_KSET_NOT_EXISTENT

Der angegebene Keyset existiert nicht.

KC_SC_INVALID_PRINCIPAL (nur auf BS2000-Systemen)

Fehler beim Anmelden mit Principal.

Returncodes bei obj_type = KC_CLUSTER_PAR:

Maincode = KC_MC_REJECTED Der Aufruf wurde von UTM abgewiesen. **Subcodes:**
KC_SC_CCFG_NO_CLUSTER_APPLI Es handelt sich nicht um eine UTM-Cluster-Anwendung.
KC_SC_CCFG_RT_CODE_NOT_OK Modifikation wurde nicht durchgeführt. UTM-interner Fehler. Bitte wenden Sie sich an die Systembetreuung.
KC_SC_CCFG_FILE_LOCK_ERROR Cluster-Konfigurationsdatei ist gesperrt.
KC_SC_CCFG_FILE_WRITE_ERROR Fehler beim Schreiben der Cluster-Konfigurationsdatei.
KC_SC_CCFG_FILE_READ_ERROR Fehler beim Lesen der Cluster-Konfigurationsdatei.
KC_SC_INVALID_BUFFER_LTH UTM-interner Fehler. Bitte wenden Sie sich an die Systembetreuung.
KC_SC_CCFG_FILE_NOT_OPEN UTM-interner Fehler. Bitte wenden Sie sich an die Systembetreuung.

Returncodes bei obj_type = KC_DIAG_AND_ACCOUNT:

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen.

Subcodes:

KC_SC_KDCMON_ERROR

Mögliche Fehlerursachen:

- Das Subsystem KDCMON wurde nicht gestartet
- Der Messmonitor KDCMON wurde nicht gestartet oder wurde inzwischen wieder beendet.

KC_SC_NOT_GEN

Der OSI-Trace soll eingeschaltet werden, obwohl keine Objekte für die verteilte Verarbeitung über OSI TP generiert sind.

KC_SC_SYSPROT_SWITCH_RUNNING

Der Umschalt-Vorgang von einer Protokolldatei auf die nächste läuft gerade. Deshalb kann aktuell kein neues Umschaltkommando ausgeführt werden.

KC_SC_TRCFILE_HANDLING_RUNNING

Es werden gerade Trace-Dateien geöffnet oder geschlossen, so dass zurzeit keine Modifikationen an den Trace-Einstellungen möglich sind.

Returncodes bei obj_type = KC_MAX_PAR:

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen.

Subcodes:

KC_SC_NOT_GEN

Die Datenlieferung an openSM2 wurde nicht generiert, d.h. sie kann nicht ein- bzw. ausgeschaltet werden.

KC_SC_NOT_AVAILABLE

openSM2 ist z. Zt. nicht verfügbar.

KC_SC_NOT_ALLOWED

Beim Ändern von *destadm* (Empfänger der Ergebnisse von KDCADM-Asynchron-TACs) wurde ein ungültiges Ziel angegeben. Mögliche Ursachen:

- In *destadm* wurde ein LTERM-Partner angegeben, der gesperrt oder gelöscht ist.
- In *destadm* wurde ein Transaktionscode angegeben, der gesperrt oder gelöscht ist.
- In *destadm* wurde ein Dialog-TAC angegeben, als Empfänger darf jedoch nur ein Asynchron-TAC oder ein LTERM-Partner angegeben werden.
- In *destadm* wurde ein LTERM-Partner angegeben, dem ein Client vom Typ UPIC-... zugeordnet ist.

KC_SC_NOT_EXISTENT

Ungültige Angabe in *destadm*. Der angegebene Name gehört weder zu einem LTERM-Partner noch zu einem Transaktionscode.

Returncodes bei obj_type = KC_TASKS_PAR:

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen.

Subcode:

KC_SC_NOT_ALLOWED

- Die in *mod_max_tasks*, *mod_max_asyntasks* oder *mod_max_tasks_in_pgwt* angegebene Prozesszahl ist größer als der in der KDCDEF-Anweisung MAX generierte Wert.
- *mod_max_tasks_in_pgwt*='0' ist nicht erlaubt, da in der Anwendung blockierende Aufrufe erlaubt sind, d.h. es sind Transaktionscodes oder TAC-Klassen mit *pgwt*='Y' generiert.

Returncodes bei obj_type = KC_TIMER_PAR:

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen.

Subcode:

KC_SC_NO_UTMD

Es wurde versucht, einen Timer für die verteilte Verarbeitung über LU6.1 oder OSI TP zu setzen, obwohl keine Objekte für die verteilte Verarbeitung generiert sind.

11.2.10 KC_ONLINE_IMPORT - Anwendungsdaten online importieren

In einer UTM-Cluster-Anwendung (Unix-, Linux- und Windows-Systeme) kann nach normaler Beendigung einer Knoten-Anwendung eine andere, laufende Knoten-Anwendung Nachrichten an LTERMs, LPAPs, OSI-LPAPs, Asynchron-TACs, TAC-Queues und offene Asynchron-Vorgänge aus der beendeten Knoten-Anwendung importieren, wenn ihre KDCFILE aus demselben Generierungslauf stammt. Importierte Daten werden in der beendeten Knoten-Anwendung gelöscht. Vor dem Importieren wird überprüft, ob bereits ein Online-Import läuft, in diesem Fall wird ein erneuter Import abgewiesen. Ein Online-Import ist nur in UTM-S-Anwendungen möglich. Offene Asynchron-Vorgänge werden nicht importiert, wenn der Vorgang Datenbank-Transaktionen mit SESAM /SQL enthält.

Ablauf / Wirkungsdauer / Transaktionssicherung / Cluster

KC_ONLINE_IMPORT stößt den Online-Import der Anwendungsdaten an, d.h. ein Auftrag zum Online-Import wird erzeugt. Bei Rückkehr in das Teilprogramm ist der Online-Import noch nicht durchgeführt. Der Online-Import unterliegt nicht der Transaktionssicherung. Er kann nicht durch einen in derselben Transaktion folgenden RSET-Aufruf zurückgesetzt werden. Der Online-Import wird von einem Prozess der Anwendung durchgeführt.

Nach der Bearbeitung des Auftrags informiert UTM Sie mit einer Meldung über Erfolg bzw. Misserfolg des Online-Imports. Wenn der Import erfolgreich war, aber wegen eines vorübergehenden Betriebsmittel-Engpass nicht alle Daten importiert werden konnten, können die nicht importierten Daten durch einen erneuten Online-Import in eine andere Knoten-Anwendung oder nach Auflösung des Engpasses in dieselbe Knoten-Anwendung importiert werden.

Diese Funktion ist nur bei Cluster-Betrieb erlaubt. Der Online-Import wird in der Knoten-Anwendung durchgeführt, in der der Aufruf erfolgt.

Versorgung der Parameter	
Parameterbereich	
Feldname	Feldinhalt
version	KC_ADMI_VERSION_1
retcode	KC_RC_NIL
version_data	KC_VERSION_DATA_11
opcode	KC_ONLINE_IMPORT
subopcode1	KC_ALL
id_lth	0
select_lth	0
data_lth	Länge der Datenstruktur
Identifikationsbereich	
—	
Selektionsbereich	
—	
Datenbereich	
Datenstruktur	

Rückgaben von UTM	
Parameterbereich	
Feldname	Inhalt
retcode	Returncodes

subopcode1

Mit *subopcode1*=KC_ALL geben Sie an, dass alle Nachrichten, also Nachrichten an (OSI-)LPAPs, an Asynchron-TACs, TAC-Queues und offene Asynchron-Vorgänge importiert werden.

data_lth

Im Feld *data_lth* geben Sie die Länge der Datenstruktur im Datenbereich an.

Datenbereich

Im Datenbereich geben Sie die Datenstruktur *kc_online_import_str* an.

In *kc_online_import_str* geben Sie die Nummer des Knotens an, von dem die Anwendungsdaten importiert werden sollen.

Die Datenstruktur *kc_online_import_str* ist folgendermaßen definiert.

struct kc_online_import_str
`char import_node[4];`

Das Feld der Datenstruktur hat die folgende Bedeutung:

import_node

Nummer des Knotens, von dem die Anwendungsdaten importiert werden sollen.

retcode

Im Feld *retcode* liefert openUTM den Returncode des Aufrufs zurück. Neben den im Abschnitt „Returncodes" aufgelisteten Returncodes können zusätzlich folgende Returncodes auftreten

Maincode = KC_MC_REJECTED Der Aufruf wurde von openUTM abgewiesen. **Subcode:**
KC_ONLINE_IMPORT_RUNNING Es wird versucht, einen Online-Import zu starten, während bereits ein Online-Import läuft.
KC_SC_CCFG_INVALID_NODE_INDEX Die Nummer der Knoten-Anwendung, von der die Anwendungsdaten importiert werden sollen, ist ungültig. Entweder ist es die Nummer der eigenen Knoten-Anwendung oder eine Nummer, die nicht zur UTM-Cluster-Anwendung gehört.
KC_SC_CCFG_INVALID_NODE_STATE Die Knoten-Anwendung, von der die Anwendungsdaten importiert werden sollen, hat einen für einen Online-Import ungültigen Status. Ungültiger Status bedeutet, dass die Knoten-Anwendung • entweder noch nie gestartet wurde • oder abnormal beendet wurde • oder noch läuft.

Maincode = KC_MC_NOT_EXISTENT Die Nummer der Knoten-Anwendung, von der importiert werden soll, liegt außerhalb des gültigen Wertebereichs 1 bis 32. **Subcode:**
KC_SC_NO_INFO

11.2.11 KC_PTC_TA - Transaktion im Zustand PTC zurücksetzen

KC_PTC_TA setzt eine Transaktion zurück, die sich im Zustand PTC (prepare to commit) befindet.

Die Identifikationsdaten der Transaktion setzen sich zusammen aus einem Tripel bestehend aus Vorgangs-Index, Vorgangs-Nummer und Transaktionsnummer. Diese Daten ermitteln Sie mit einem vorhergehenden Aufruf KC_GET_OBJECT, Operationscode KC_PTC.

Ablauf / Wirkungsdauer / Transaktionssicherung / Cluster

Durch diesen Aufruf wird der lokale Teil einer verteilten Transaktion zurückgesetzt.
Die Verteilte Transaktion kann administrativ nicht zurückgesetzt werden, sondern immer nur der lokale Teil einer solchen Transaktion. Ein solches administrative Rücksetzen ist eine heuristische Entscheidung über den Ausgang der Transaktion und **kann ggf. zu Inkonsistenzen in den verteilten Datenbeständen führen**, wenn die Verteilte Transaktion durch den Commit Coordinator vorgesetzt wird.

Versorgung der Parameter	
Parameterbereich	
Feldname	Feldinhalt
version	KC_ADMI_VERSION_1
retcode	KC_RC_NIL
version_data	KC_VERSION_DATA_11
opcode	KC_PTC_TA
subopcode1	KC_ROLLBACK
id_lth	25
select_lth	0
data_lth	0
Identifikationsbereich	
Tripel mit den Identifikationsdaten der Transaktion	
Selektionsbereich	
—	
Datenbereich	
—	

Rückgaben von UTM	
Parameterbereich	
Feldname	Inhalt
retcode	Returncodes

subopcode1

Mit *subopcode1*=KC_ROLLBACK geben Sie an, dass die Transaktion zurückgesetzt werden soll.

id_lth

Im Feld *id_lth* geben Sie die Länge der Datenstruktur *kc_ptc_id_str* an.

Identifikationsbereich

Im Identifikationsbereich geben Sie die Datenstruktur *kc_pct_id_str* an.

kc_ptc_id_str muss mit den Werten gefüllt werden, die beim Aufruf KC_GET_OBJECT mit Operationscode KC_PTC in der Struktur *ptc_ident* zurückgegeben wurden. *ptc_ident* ist in der Datenstruktur *kc_ptc_str* enthalten, siehe Abschnitt "kc_ptc_str - Transaktionen im Zustand PTC".

Die Datenstruktur *kc_ptc_id_str* ist folgendermaßen definiert.

struct kc_ptc_id_str

```
char vg_indx[10];
char vg_nr[10];
char ta_nr_in_vg[5];
```

vg_indx ist der Index des Vorgangs, *vg_nr* die Nummer des Vorgangs und *ta_nr_in_vg* die Nummer der Transaktion im Vorgang.

retcode

Im Feld *retcode* liefert openUTM den Returncode des Aufrufs zurück. Neben den im Abschnitt „Returncodes" aufgelisteten Returncodes können zusätzlich folgende Returncodes auftreten

Maincode = KC_MC_REJECTED Der Aufruf wurde von openUTM abgewiesen. **Subcode:**
KC_SC_NO_MORE_PTC Die Transaktion ist nicht mehr im Zustand PTC.
KC_SC_END_TA_ALREADY_INITIATED Das Beenden der Transaktion wurde schon initiiert. Dies kann folgende Gründe haben:

Maincode = KC_MC_REJECTED Der Aufruf wurde von openUTM abgewiesen. **Subcode:** • Der Partner der Verteilten Transaktion, der über das Ergebnis der Transaktion entscheidet (Commit Coordinator), hat das Beenden der Transaktion initiiert • Das Beenden der Transaktion wurde per Administration initiiert.
KC_SC_PARTNER_CONNECTED Die Verbindung zu dem Partner der Verteilten Transaktion, der über das Ergebnis der Transaktion entscheidet (Commit Coordinator), ist aufgebaut. Dadurch wird das Beenden der Transaktion initiiert.

11.2.12 KC_SEND_MESSAGE - Nachricht senden (BS2000-Systeme)

Mit KC_SEND_MESSAGE können Sie eine Nachricht an ein, mehrere oder alle aktiven Terminals einer UTM-Anwendung auf einem BS2000-System senden. Der Text der Nachricht darf bis zu 74 Zeichen lang sein und wird im Datenbereich an UTM übergeben. Als Nachricht sendet UTM dann die Meldung K023 mit der angegebenen Nachricht als Insert. Sie wird standardmäßig in der Systemzeile am Terminal ausgegeben. Das Meldungsziel der Meldung K023 kann jedoch auch geändert werden. Ist für die UTM-Meldung K023 das Meldungsziel PARTNER ausgewählt (siehe openUTM-Handbuch „Meldungen, Test und Diagnose auf BS2000-Systemen“), können Sie die Nachricht auch an eine, mehrere oder alle konnektierten TS-Anwendungen senden. Die Nachricht geht nur an Dialog-Partner (LTERM mit USAGE=D).

Mit KC_SEND_MESSAGE können Sie:

- eine Nachricht an alle Terminals senden, die zur Zeit mit der Anwendung verbunden sind. Dies gilt auch für Terminals, die über einen LTERM-Pool mit der Anwendung verbunden sind.
- eine Nachricht an alle TS-Anwendungen senden, die mit der UTM-Anwendung verbunden sind, sofern das Meldungsziel PARTNER für K023 generiert ist.
- eine Nachricht an einen bestimmten Terminal-Benutzer oder, vorausgesetzt das Meldungsziel PARTNER ist generiert, an eine bestimmte TS-Anwendung senden. In diesem Fall müssen Sie im Identifikationsbereich den Namen des LTERM-Partners angeben, über den das Terminal mit der Anwendung verbunden ist. Voraussetzung für das Zustellen der Nachricht ist, dass das Terminal zum Zeitpunkt des KC_SEND_MESSAGE-Aufrufs mit der Anwendung verbunden ist.

  Wollen Sie die Nachricht an einen bestimmten Benutzer senden, dann können Sie den LTERM-Partner, über den der Benutzer bei der Anwendung angemeldet ist, wie folgt ermitteln:

  Sie fordern mit KC_GET_OBJECT zunächst Informationen über die Benutzerkennung an, mit der sich der Benutzer bei der Anwendung angemeldet hat (Objekttyp KC_USER).

  UTM liefert dann die Eigenschaften der Benutzerkennung in der Datenstruktur *kc_user_str* zurück. Ist der Benutzer zum Zeitpunkt der Abfrage mit der Anwendung verbunden, dann steht im Feld *lterm_curr* der Name des LTERM-Partners, über den der Benutzer angemeldet ist. Diesen Namen übergeben Sie beim Senden der Nachricht mit KC_SEND_MESSAGE im Identifikationsbereich.

Ablauf / Transaktionssicherung

Ein Aufruf von KC_SEND_MESSAGE unterliegt nicht der Transaktionssicherung. Er ist nicht durch einen RSET-Aufruf in derselben Transaktion rücksetzbar.

Geben Sie im Identifikationsbereich keinen Empfänger an und setzen Sie im Parameterbereich *obj_number*=0, dann ermittelt UTM alle derzeit aktiven LTERM-Partner, die mit *usage_type*='D' eingetragen sind, und sendet ihnen die Nachricht zu. Bei der Rückkehr in das Teilprogramm wurde die Nachricht bereits gesendet.

Geben Sie im Identifikationsbereich den Namen eines LTERM-Partners an und setzen im Parameterbereich *obj_number*=1, dann bedeutet die erfolgreiche Bearbeitung des KC_SEND_MESSAGE-Aufrufs, dass die Nachricht an diesen LTERM-Partner gesendet wurde. Ist der LTERM-Partner zur Zeit nicht erreichbar, dann liefert UTM einen entsprechenden Returncode zurück.

KDCSEND ("KDCSEND - Nachricht an LTERM-Partner senden (BS2000- Systeme)")

Versorgung der zu übergebenden Bereiche

Funktion des Aufrufs	Angabe im			
	Parameterbereich [1]	Identifikationsbereich	Selektionsbereich	Datenbereich
Nachricht an alle aktiven LTERM-Partner senden	*obj_number*: 0	——	——	Nachricht
Nachricht an einen LTERM-Partner senden	*obj_number*: 1	Name des LTERM-Partners	——	Nachricht

[1] In allen Fällen muss im Parameterbereich der Operationscode KC_SEND_MESSAGE angegeben werden.

Versorgung der Parameter	
Parameterbereich	
Feldname	Inhalt
version	KC_ADMI_VERSION_1
retcode	KC_RC_NIL
version_data	KC_VERSION_DATA_11
opcode	KC_SEND_MESSAGE
obj_number	1 / 0
id_lth	Länge Objektname / 0
select_lth	0
data_lth	Länge der Nachricht
Identifikationsbereich	
Objektname / —	
Selektionsbereich	
—	
Datenbereich	
Nachricht	

KDCADMI-Aufruf
KDCADMI (&parameter_area, &identification_area, NULL, &data_area) oder KDCADMI (&parameter_area, NULL, NULL, &data_area)

Rückgaben von UTM	
Parameterbereich	
Feldname	Inhalt
retcode	Returncodes

obj_number

In *obj_number* geben Sie an, ob die Nachricht an alle derzeit aktiven LTERM-Partner gesendet werden soll, oder nur an einen bestimmten LTERM-Partner.

- *obj_number*=0 bedeutet:
  Die Nachricht soll an alle aktiven LTERM-Partner gesendet werden. Als Adresse des Identifikationsbereichs muss der Nullpointer übergeben werden.
- *obj_number*=1 bedeutet:
  Die Nachricht soll nur an einen LTERM-Partner gesendet werden. Der Name des LTERM-Partners ist im Identifikationsbereich zu übergeben.

id_lth

In *id_lth* ist die Länge des Identifikationsbereichs anzugeben. D.h.:

- bei *obj_number*=0 müssen Sie *id_lth*=0 setzen.
- bei *obj_number*=1 müssen Sie in *id_lth* die Länge des Objektnamens angeben, der im Identifikationsbereich übergeben wird.

data_lth

Länge der Nachricht, die gesendet werden soll. Die Nachricht müssen Sie im Datenbereich übergeben. Es muss sein: 1 <= *data_lth* <= 74.

Identifikationsbereich

Wie Sie den Identifikationsbereich versorgen müssen, ist abhängig von *obj_number*.

- bei *obj_number*=0 müssen Sie beim KC_SEND_MESSAGE-Aufruf den Nullpointer übergeben.
- bei *obj_number*=1 müssen Sie im Identifikationsbereich die Union *kc_id_area* mit dem Namen des LTERM-Partners angeben (Feld *kc_name8*), an den die Nachricht gesendet werden soll.

Datenbereich

Im Datenbereich ist die Nachricht, die UTM senden soll, als Character-String zu übergeben. Die Nachricht darf nicht länger als 74 Zeichen sein.

retcode

Im Feld *retcode* liefert UTM den Returncode des Aufrufs zurück. Neben den im Abschnitt „Returncodes" aufgelissteten Returncodes können zusätzlich folgende Returncodes auftreten.

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen.

Subcodes:

KC_SC_NOT_EXISTENT

Der im Identifikationsbereich angegebene Name ist unbekannt, ein LTERM-Partner mit diesem Namen existiert nicht.

KC_SC_NOT_ALLOWED

Für den im Identifikationsbereich angegebenen LTERM-Partner bzw. für den Client, der diesem LTERM-Partner zugeordnet ist, ist die Operation nicht zulässig.
Mögliche Gründe für die Ablehnung sind:

- Es besteht derzeit keine Verbindung zu dem Client; der LTERM-Partner ist nicht aktiv.
- Dem LTERM-Partner ist derzeit kein Client zugeordnet.
- Der angegebene LTERM-Partner ist kein Dialog-Partner, d.h. er wurde mit *usage_type*='O' konfiguriert.
- Der Client, der dem angegebenen LTERM-Partner zugeordnet ist, wurde aus der Konfiguration gelöscht.

KC_SC_DELETED

Der angegebene LTERM-Partner existiert nicht mehr, er wurde aus der Konfiguration der Anwendung gelöscht.

11.2.13 KC_SHUTDOWN - Anwendungslauf beenden

Mit KC_SHUTDOWN können Sie den aktuellen Anwendungslauf beenden.

In UTM-Cluster-Anwendungen (Unix-, Linux- und Windows-Systeme) können Sie angeben, ob der Anwendungslauf auf allen Knoten beendet wird oder nur auf dem Knoten, an dem der Aufruf erfolgt.

Folgende Möglichkeiten stehen Ihnen zur Verfügung:

- Sie können den Anwendungslauf normal beenden. UTM beendet den Anwendungslauf, sobald alle laufenden Dialog-Schritte beendet sind (KC_NORMAL).
- Sie können den Anwendungslauf zeitgesteuert nach einer angegebenen Zeitspanne beenden (KC_WARN).
- Sie können den Anwendungslauf beenden, nachdem alle UTM-D-Dialoge beendet und alle UTM-D-Verbindungen abgebaut sind, spätestens jedoch nach einer angegeben Zeitspanne (KC_GRACEFUL).
- Sie können den Anwendungslauf abbrechen, d.h. sofort abnormal beenden (KC_KILL).

Zum Beenden eines UTM-Anwendungslaufs siehe auch die openUTM-Handbuch „Einsatz von UTM-Anwendungen“.

Beim Anwendungsabbruch ist Folgendes zu beachten:

Der Anwendungsabbruch (KC_KILL) kann nicht als Asynchron-Vorgang erfolgen, er ist nur im Dialog erlaubt. Der Aufruf mit *subopcode1*=KC_KILL in einem Asynchron-Vorgang wird von UTM abgewiesen.

Beim Shutdown von Anwendungen mit verteilter Verarbeitung ist Folgendes zu beachten:

Anwendungen mit verteilter Verarbeitung sollten Sie am besten mit KC_GRACEFUL beenden, alternativ mit KC_WARN. Dabei sollten Sie eine Zeit angeben, die größer ist als die Zeit, die eine verteilte Transaktion maximal im Zustand PTC (d.h. Transaktionsstatus P) verbleibt. Damit verringert sich die Wahrscheinlichkeit, dass verteilte Transaktionen beim Anwendungsende noch in diesem Zustand sind und die Anwendung abnormal mit END-PET beendet wird.

Generell gilt:
Eine Anwendung mit verteilter Verarbeitung wird nicht normal beendet, wenn es zum Zeitpunkt des Shutdowns noch Vorgänge mit Transaktionsstatus P 'preliminary end of transaction' gibt, oder wenn für Asynchron-Nachrichten an einen Partner-Server noch keine Quittungen eingetroffen sind. UTM gibt dann die Meldung K060 mit der Abbruchursache ENDPET aus. Es werden keine Dumps erzeugt.

Ablauf / Wirkungsdauer / Transaktionssicherung / Cluster:

Der Aufruf von KC_SHUTDOWN unterliegt nicht der Transaktionssicherung. Er ist nicht durch einen RSET-Aufruf rücksetzbar.

Der Abbruch eines Anwendungslaufs (KC_KILL) wirkt sofort, es wird nicht mehr in das Teilprogramm zurückgekehrt.

Soll die Anwendung beendet werden (KC_NORMAL, KC_WARN und KC_GRACEFUL), dann wird durch den Aufruf ein Auftrag erzeugt, d.h. Aktionen zum Shutdown angestoßen.

Der Ablauf des Shutdown, d.h. wie und wann UTM den Anwendungslauf beendet, ist abhängig von den Angaben für *subopcode1* im Parameterbereich. Der Ablauf des Shutdown wird unter Punkt KC_GRACEFUL beschrieben.

Für UTM-Cluster-Anwendungen (Unix-, Linux- und Windows-Systeme) gilt:
Der Aufruf kann sowohl Cluster-global als auch Knoten-lokal wirken, d.h. der aktuelle Anwendungslaufs wird auf allen Knoten beendet, oder der Anwendungslauf wird nur auf dem Knoten beendet, auf dem der Aufruf erfolgte.

KDCSHUT ("KDCSHUT - Anwendungslauf beenden")

Versorgung der zu übergebenden Bereiche

Funktion des Aufrufs	Angabe im			
	Parameterbereich [1]	Identifikationsbereich	Selektionsbereich	Datenbereich
Anwendungslauf sofort abbrechen (nur im Dialog)	*subopcode1*: KC_KILL	——	——	—— bzw. *kc_shutdown_str*
Anwendungslauf normal beenden	*subopcode1*: KC_NORMAL	——	——	—— bzw. *kc_shutdown_str*
Anwendungslauf nach Ablauf eines Timers normal beenden openUTM auf einem BS2000-System gibt Standardmeldung an alle aktiven Benutzer aus	*subopcode1*: KC_WARN	——	——	*kc_shutdown_str*
Anwendungslauf auf einem BS2000-Systeme nach Ablauf eines Timers normal beenden und eine Meldung an alle aktiven Benutzer senden	*subopcode1* : KC_WARN, *subopcode2* : KC_USER_MSG	——	——	*kc_shutdown_str*
Anwendungslauf normal beenden, wenn alle UTM- D-Verbindungen abgebaut sind, spätestens aber nach Ablauf des Timers	*subopcode1* : KC_GRACEFUL	——	——	*kc_shutdown_str*

[1] In allen Fällen muss im Parameterbereich der Operationscode KC_SHUTDOWN angegeben werden.

Versorgung der Parameter	
Parameterbereich	
Feldname	Inhalt
version	KC_ADMI_VERSION_1
retcode	KC_RC_NIL
version_data	KC_VERSION_DATA_11
opcode	KC_SHUTDOWN
subopcode1	KC_GRACEFUL / KC_KILL / KC_NORMAL / KC_WARN
subopcode2	KC_USER_MSG / —
id_lth	0
select_lth	0
data_lth	Länge der Daten im Datenbereich / 0
Identifikationsbereich	
—	
Selektionsbereich	
—	
Datenbereich	
Datenstruktur kc_shutdown_str / —	

KDCADMI-Aufruf
KDCADMI (&parameter_area, NULL, NULL, &data_area) oder KDCADMI (&parameter_area, NULL, NULL, NULL)

Rückgaben von UTM	
Parameterbereich	
Feldname	Inhalt
retcode	Returncodes

subopcode1

In *subopcode1* geben Sie an, wie UTM die Anwendung beenden soll. Folgende Angaben sind möglich:

KC_GRACEFUL

UTM bereitet den Shutdown vor. Die Anwendung wird beendet, sobald alle UTM-D-Dialoge beendet und alle UTM-D-Verbindungen abgebaut sind, spätestens jedoch nach Ablauf des angegebenen Timers. Den Wert des Timers müssen Sie im Datenbereich übergeben.

Ist der angegebene Timer abgelaufen, wird die Anwendung auf jeden Fall beendet. Bestehen keine UTM-D-Verbindungen, wird die Anwendung sofort normal beendet.

Nach Bearbeitung des KC_GRACEFUL-Aufrufs gilt Folgendes:

- Es können sich nur noch Benutzer mit Administrationsberechtigung anmelden. Anmeldeversuche anderer Benutzer werden abgelehnt.
- Es können nur noch Transaktionscodes von Administrationsprogrammen und die UTM-Benutzerkommandos außer KDCOUT aufgerufen werden. Alle anderen Vorgänge werden von UTM nicht mehr gestartet.
- Alle aktiven Verbindungen zu LPAP- und OSI-LPAP-Partnern werden auf QUIET gesetzt.

KC_KILL Der Anwendungslauf wird abgebrochen, d.h. sofort beendet. Offene Vorgänge werden nicht mehr beendet. Von allen Prozessen wird ein UTM-Dump erstellt mit REASON='ASIS99'.

KC_NORMAL

Der Anwendungslauf wird normal beendet.

Der Shutdown wird sofort eingeleitet. Nach dem KC_SHUTDOWN-Aufruf gilt Folgendes:

- Benutzer/Clients können sich nicht mehr bei der Anwendung anmelden.
- Aufträge von Partner-Servern werden nicht mehr angenommen. Bereits angemeldete Benutzer /Clients können keine neuen Vorgänge mehr starten.
- Neue Dialog-Eingaben werden nicht mehr bearbeitet. Ist die neue Dialog-Eingabe Teil einer Mehrschritt-Transaktion, dann wird die Mehrschritt-Transaktion auf den letzten Sicherungspunkt zurückgesetzt.
- Alle logischen Verbindungen zu Clients, Druckern und Partner-Anwendungen werden abgebaut.

Offene Vorgänge können nach dem nächsten Start der Anwendung weiter bearbeitet werden.

KC_WARN

UTM bereitet den Shutdown vor. Die Anwendung wird nach Ablauf des angegebenen Timers beendet. Den Wert des Timers müssen Sie im Datenbereich übergeben. Nach Bearbeitung des KC_SHUTDOWN-Aufrufs gilt Folgendes:

- Es können sich nur noch Benutzer mit Administrationsberechtigung anmelden. Anmeldeversuche anderer Benutzer werden abgelehnt.
- Es können nur noch Transaktionscodes von Administrationsprogrammen und die UTM-Benutzerkommandos außer KDCOUT aufgerufen werden. Alle anderen Vorgänge werden von UTM nicht mehr gestartet.
- Alle aktiven Verbindungen zu LPAP- und OSI-LPAP-Partnern werden auf QUIET gesetzt.

subopcode2

subopcode2 ist nur relevant, wenn Sie *subopcode1*=KC_WARN angeben. In allen anderen Fällen darf in *subopcode2* nichts angegeben werden.

Sie geben *subopcode2*=KC_USER_MSG an, wenn UTM zur Vorbereitung auf den Shutdown eine Nachricht an alle derzeit aktiven Benutzer senden soll. Die Nachricht, die UTM senden soll, müssen Sie im Datenbereich übergeben.

In UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen wird die Nachricht akzeptiert, es werden aber keine Warnmeldungen ausgegeben.

Wenn Sie auf BS2000-Systeme *subopcode2* bei KC_WARN nicht angeben, dann werden alle aktiven Benutzer durch eine Standardmeldung über den bevorstehenden Shutdown und die bis dahin verbleibende Zeit informiert.

data_lth

Im Feld *data_lth* geben Sie die Länge des an UTM übergebenen Datenbereichs an.

- bei *subopcode1*=KC_KILL, KC_NORMAL:
  Im Datenbereich werden keine Daten an/von UTM übergeben (*data_lth*='0'), oder oder die Länge der Datenstruktur *kc_shutdown_str*, die Sie im Datenbereich übergeben.
- bei *subopcode1*=KC_GRACEFUL, KC_WARN:
  Im Feld *data_lth* geben Sie die Länge der Datenstruktur *kc_shutdown_str* an, die Sie im Datenbereich an UTM übergeben.

Datenbereich

Im Datenbereich müssen Sie bei *subopcode1*=KC_GRACEFUL und *subopcode1*=KC_WARN die Datenstruktur *kc_shutdown_str* an UTM übergeben. *kc_shutdown_str* muss die Größe des Timers enthalten und wenn zusätzlich *subopcode2*=KC_USER_MSG ist, die Nachricht, die als Warnung an alle Terminal-Benutzer gesendet werden soll. Die Datenstruktur *kc_shutdown_str* wird in der Include-Datei *kcadminc.h* zur Verfügung gestellt.

Bei stand-alone UTM-Anwendungen muss der Datenbereich nur für KC_WARN und KC_GRACEFUL versorgt werden. Das Feld *scope* in *kc_shutdown_str* wird nicht ausgewertet.

In UTM-Cluster-Anwendungen gilt:
Bei jedem *subopcode1*: In der Datenstruktur *kc_shutdown_str* können Sie mit dem Feld *scope* steuern, ob nur die lokale Knoten-Anwendung beendet werden soll, oder ob Sie die gesamte UTM-Cluster-Anwendung, also alle Knoten-Anwendungen, beenden möchten. Für einen globalen Shutdown der UTM-Cluster-Anwendung müssen Sie die Datenstruktur *kc_shutdown_str* mit *scope*='G' versorgen. Wenn Sie im Cluster keine Datenstruktur angeben, wird ein lokaler Shutdown ausgeführt.

Die Datenstruktur ist wie folgt aufgebaut:

struct kc_shutdown_str

`char time_min[3];`
`char user_message[74];`
`char scope;`

time_min

In *time_min* geben Sie die Zeit in Minuten an, nach der UTM den Anwendungslauf normal beenden soll.

Sie sollten eine Zeit angeben, die größer ist als die Zeit, die eine verteilte Transaktion maximal im Zustand PTC (d.h. Transaktionsstatus P) verbleibt. Dies ist in Auftragnehmer-Vorgängen die mit MAX PTCTIME generierte Zeit und in LU6.1-Auftraggeber-Vorgängen die beim verwendeten LTAC generierte Zeit *time2* des Operanden WAITTIME.

Minimalwert: '1'

Maximalwert: '255'

Die Angabe *time_min*='0' wird von UTM abgelehnt. Soll die Anwendung ohne Verzögerung normal beendet werden, dann müssen Sie *subop code1*=KC_NORMAL angeben.

Besonderheiten bei UTM-Anwendungen auf BS2000-Systemen

- *time_min* wird immer zusammen mit der Shutdown-Ankündigung an den aktiven Terminals ausgegeben.
- Bei großen UTM-Anwendungen auf BS2000-Systemen (Konfigurationen mit vielen Clients) benötigt UTM für die Ausgabe der Shutdown-Ankündigung eine gewisse Zeit. Deshalb sollten Sie *time_min* nicht zu klein wählen.
- Zusätzlich sollten Sie für den Transaktionscode, über den das Teilprogramm mit diesem KC_SHUTDOWN-Aufruf gestartet wird, einen hinreichend großen Wert für *cpu_time_msec* definieren bzw. die CPU-Zeit nicht überwachen lassen (siehe *kc_tac_str* im Abschnitt "kc_tac_str - Transaktionscodes lokaler Services"). *cpu_time_msec* gibt die CPU-Zeit an, die der Teilprogrammlauf maximal verbrauchen darf. Wird die Zeit zu klein gewählt, dann kann es passieren, dass der Shutdown abgebrochen wird.

user_message

Ist nur bei *subopcode2*=KC_USER_MESSAGE relevant. Wurde kein *subopcode2* angegeben, dann wird der Bereich ignoriert.

In *user_message* können Sie eine eigene Nachricht übergeben, die UTM vor dem Shutdown als Warnung an alle Terminal-Benutzer senden soll. Die Nachricht darf maximal 74 Zeichen lang sein.

openUTM auf BS2000-Systemen

- Übergeben Sie in *user_message* keine eigene Warn-Nachricht, dann gibt UTM an alle Terminal-Benutzer, die derzeit mit der Anwendung verbunden sind, die Meldung K023 mit den folgenden Inserts aus:

- 'hour':'minutes':'seconds'

APPLICATION 'name' WILL BE TERMINATED IN 'minutes' MINUTES

openUTM auf Unix-, Linux- und Windows-Systemen

- Auf Unix-, Linux- und Windows-Systemen werden keine Warnmeldungen ausgegeben.

scope steuert, ob nur die lokale Knoten-Anwendung beendet wird, oder ob die gesamte UTM-Cluster-Anwendung, also alle Knoten-Anwendungen, beendet wird. Das Feld *scope* wird nur für UTM-Cluster-Anwendungen ausgewertet.

'L' Nur die lokale Knoten-Anwendung wird beendet.

'G' Alle Knoten-Anwendungen des Clusters und damit die gesamte UTM-Cluster-Anwendung werden beendet.

retcode

Im Feld *retcode* liefert UTM den Returncode des Aufrufs zurück. Neben den im Abschnitt „Returncodes" aufgelisteten Returncodes können zusätzlich folgende Returncodes auftreten.

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen.

Subcode:

KC_SC_NOT_ALLOWED

subopcode1=KC_KILL ist in einem Asynchron-Vorgang verwendet worden.

KC_SC_NO_GLOB_CHANG_POSSIBLE

Die Generierung der Knoten-Anwendungen ist zur Zeit nicht konsistent. Zuerst sollten die Knoten-Anwendungen mit alter Generierung beendet werden.

Maincode = KC_MC_DATA_INVALID

Ein Feld der Datenstruktur im Datenbereich enthält einen ungültigen Wert.

Subcode:

KC_SC_INVALID_MOD

Nur bei *subopcode1*=KC_GRACEFUL und *subopcode1*=KC_WARN:
Der Anwendungslauf wurde nicht beendet, weil die Zeitangabe in *time_min* ungültig ist.

Maincode = KC_MC_REJECTED_CURR

Der Aufruf kann zur Zeit nicht bearbeitet werden.

Subcode:

KC_SC_INVDEF_RUNNING

Nur bei UTM-Cluster-Anwendungen:
Es läuft gerade ein inverser KDCDEF, d.h. der Auftrag kann z.Zt. nicht bearbeitet werden.

Maincode = KC_MC_RECBUF_FULL

Nur bei UTM-Cluster-Anwendungen:

Subcode:

KC_SC_NO_INFO

Der Puffer mit Wiederanlauf-Information ist voll (siehe openUTM-Handbuch „Anwendungen generieren “, KDCDEF-Steueranweisung MAX, Parameter RECBUF).

11.2.14 KC_SPOOLOUT - Verbindungen zu Druckern aufbauen

Mit KC_SPOOLOUT können Sie Verbindungen zu Druckern aufbauen. Sie können:

- die Verbindungen zu allen Druckern aufbauen, für die Druckaufträge in der zugehörigen Message Queue stehen, und zu denen noch keine Verbindung existiert.
- die Verbindung zu den Druckern aufbauen, die einem bestimmten LTERM-Partner zugeordnet sind. Der Name des LTERM-Partners muss im Identifikationsbereich übergeben werden.

Ablauf / Transaktionssicherung / Cluster

Der Aufruf von KC_SPOOLOUT unterliegt nicht der Transaktionssicherung. Er ist nicht durch einen RSET-Aufruf rücksetzbar.

Durch den Aufruf wird der Verbindungsaufbau angestoßen, d.h. es wird lediglich ein Auftrag erzeugt. Es kann keine Aussage darüber gemacht werden, ob und wann eine Verbindung tatsächlich zustande kommt. Den Zustand der Verbindung können Sie dann mit einer Informationsabfrage (z.B. KC_GET_OBJECT mit *obj_type*=KC_LTERM) ermitteln.

Für UTM-Cluster-Anwendungen (Unix-, Linux- und Windows-Systeme) gilt:
Der Aufruf wirkt Knoten-lokal, d.h. die Verbindungen zu den Druckern werden nur in der Knoten-Anwendung aufgebaut, an der der Aufruf erfolgt.

Dauer der Verbindung

Die Verbindungen zu Druckern, für die kein Print Level (PLEV) definiert ist, bleiben solange bestehen, bis sie explizit abgebaut werden (siehe KC_MODIFY_OBJECT) oder der Anwendungslauf beendet wird. Die Verbindungen zu Druckern, für die ein Print Level definiert ist (PLEV > 0), werden nach dem Drucken wieder abgebaut.

Mit KDCAPPL SPOOLOUT=ON ("KDCAPPL - Eigenschaften und Grenzwerte für den Betrieb ändern") können Sie die Verbindungen zu allen Druckern aufbauen, für die Druckaufträge vorliegen.

Versorgung der zu übergebenden Bereiche

Funktion des Aufrufs	Angabe im			
	Parameterbereich[1]	Identifikationsbereich	Selektionsbereich	Datenbereich
Verbindung zu einem Drucker1 bzw. den Druckern eines Druckerbündels aufbauen	*obj_number*: 1	Name des LTERM-Partners, der dem Drucker bzw. Druckerbündel zugeordnet ist	——	——
Verbindungen zu allen derzeit nicht verbundenen Druckern aufbauen, für die Druckaufträge vorliegen	*obj_number*: 0	——	——	——

[1] In allen Fällen muss im Parameterbereich der Operationscode KC_SPOOLOUT angegeben werden.

Versorgung der Parameter	
Parameterbereich	
Feldname	Inhalt
version	KC_ADMI_VERSION_1
retcode	KC_RC_NIL
version_data	KC_VERSION_DATA_11
opcode	KC_SPOOLOUT
obj_number	1 / 0
id_lth	Länge Objektname im Identifikationsbereich / 0
select_lth	0
data_lth	0
Identifikationsbereich	
Objektname / —	
Selektionsbereich	
—	
Datenbereich	
—	

KDCADMI-Aufruf
KDCADMI (&parameter_area, &identification_area, NULL, NULL) oder KDCADMI (&parameter_area, NULL, NULL, NULL)

Rückgaben von UTM	
Parameterbereich	
Feldname	Inhalt
retcode	Returncodes

obj_number

Die Angaben in *obj_number* haben folgende Bedeutung:

- *obj_number*=0:
  UTM soll die Verbindung zu allen Druckern aufbauen, zu denen derzeit keine Verbindung existiert und für die Druckaufträge vorliegen.
- *obj_number*=1:
  UTM soll die Verbindung zu dem Drucker bzw. Druckerbündel aufbauen, der/das einem bestimmten LTERM-Partner zugeordnet ist. Den Namen des LTERM-Partners müssen Sie im Identifikationsbereich übergeben.

id_lth

In *id_lth* müssen Sie die Länge des Objektnamens angeben, den Sie im Identifikationsbereich an UTM übergeben.

- bei *obj_number*=0 ist *id_lth*=0 anzugeben.
- bei *obj_number*=1 ist für *id_lth* die Länge des Namens anzugeben, der im Identifikationsbereich übergeben wird.

Identifikationsbereich

Welche Angaben Sie im Identifikationsbereich machen müssen, ist abhängig von *obj_number*:

- *obj_number*=0:
  Sie dürfen im Identifikationsbereich keinen Objektnamen angeben. Beim KDCADMI-Aufruf müssen Sie den Nullpointer übergeben.
- *obj_number*=1:
  Im Identifikationsbereich übergeben Sie den Namen des LTERM-Partners, der dem Drucker bzw. dem Druckerbündel zugeordnet ist. Dazu legen Sie die Union *kc_id_area* über den Identifikationsbereich und übergeben den Namen des LTERM-Partners im Feld *kc_name8.*

retcode

Im Feld *retcode* liefert UTM den Returncode des Aufrufs zurück. Neben den im Abschnitt „Returncodes" aufgelisteten Returncodes können zusätzlich folgende Returncodes auftreten:

Maincode = KC_MC_REJECTED Der Aufruf wurde von UTM abgewiesen. **Subcodes:**
KC_SC_NOT_EXISTENT Der im Identifikationsbereich angegebene LTERM-Partner existiert nicht.
KC_SC_NOT_ALLOWED Für den angegebenen LTERM-Partner ist die Operation nicht zulässig. Mögliche Gründe sind: • LTERM-Partner ist ein Dialog-Partner, d.h. er ist nicht für Drucker definiert (*usage_type* != 'O'). • Dem LTERM-Partner ist zur Zeit kein Drucker/Druckerbündel zugeordnet. • Der LTERM-Partner oder der zugehörige Drucker ist zur Zeit gesperrt. • Der zu dem LTERM-Partner gehörende Drucker wurde aus der Konfiguration gelöscht. • Für den angegebenen Drucker liegen keine Nachrichten vor, d.h. die Message Queue des LTERM-Partners ist leer.
KC_SC_DELETED Der angegebene LTERM-Partner wurde aus der Konfiguration gelöscht.

11.2.15 KC_SYSLOG - System-Protokolldatei administrieren

Mit KC_SYSLOG können Sie die System-Protokolldatei SYSLOG im laufenden Betrieb administrieren. Der Umfang der Funktionen, die Ihnen zur Administration der SYSLOG zur Verfügung stehen, ist abhängig davon, ob die SYSLOG als einfache Datei oder als Dateigenerationsgruppe (BS2000-Systeme) bzw. Dateigenerationsverzeichnis (Unix-, Linux- und Windows-Systeme) angelegt wurde. Für Dateigenerationsverzeichnis bzw. Dateigenerationsgruppe wird im Folgenden die Abkürzung FGG (**F**ile **G**eneration **G**roup) verwendet.

Zu SYSLOG siehe auch das openUTM-Handbuch „Anwendungen generieren“ und das jeweilige openUTM-Handbuch „Einsatz von UTM-Anwendungen“.

Folgende Funktionen stehen Ihnen zur Verfügung, unabhängig davon, ob die SYSLOG als einfache Datei oder als FGG geführt wird:

- Den Inhalt des UTM-internen Meldungspuffers in die SYSLOG schreiben.

  Diese Funktion ist nützlich, wenn die SYSLOG, die als einfache Datei angelegt wurde, im laufenden Betrieb ausgewertet werden soll. Es werden dann alle Meldungen mit dem Meldungsziel SYSLOG in der Auswertung berücksichtigt, die bis zu diesem Zeitpunkt von UTM erzeugt wurden.

  Ist die SYSLOG als FGG angelegt, dann gilt Folgendes:
  Beim Umschalten der SYSLOG auf die nächste Dateigeneration schreibt UTM den Meldungspuffer automatisch vor dem Umschalten in die „alte“ Dateigeneration.
- Sich Information über die SYSLOG-Datei anzeigen lassen.

Folgende Funktionen können Sie zusätzlich nutzen, wenn die SYSLOG als FGG angelegt ist:

- Die automatische Größenüberwachung der SYSLOG ein- und ausschalten.

  Automatische Größenüberwachung heißt: UTM schaltet die SYSLOG automatisch auf die nächste Dateigeneration der SYSLOG-FGG um, sobald die Größe der aktuellen SYSLOG-Dateigeneration einen vorgegebenen Schwellwert überschreitet.
- Den Schwellwert für die Größenüberwachung ändern.
- Die SYSLOG auf die nächste Dateigeneration der SYSLOG-FGG umschalten.

Sie können die Größenüberwachung der SYSLOG auch dann einschalten, wenn sie nicht mit KDCDEF generiert wurde.

Ablauf beim Umschalten der SYSLOG auf eine andere Dateigeneration

Vor dem Umschalten auf eine neue Dateigeneration schreibt UTM noch die im internen Meldungspuffer zwischengespeicherten Meldungen in die alte Dateigeneration. Damit werden alle Meldungen, die vor dem Umschalten erzeugt wurden, noch in die „alte“ SYSLOG geschrieben. UTM garantiert, dass Meldungen, die nach dem Umschaltzeitpunkt (erfolgreiches Ausführen des KC_SYSLOG-Aufrufs) erzeugt werden, nicht mehr in die „alte“ SYSLOG-Dateigeneration geschrieben werden.

In UTM-Anwendungen auf BS2000-Systemen ist Folgendes zu beachten:

- Nach dem Umschalten (d.h. erfolgreiche Bearbeitung des KC_SYSLOG-Aufrufs) können Sie nicht sofort über die alte Dateigeneration verfügen. Die alte Dateigeneration wird eventuell noch längere Zeit von UTM-Prozessen offengehalten. Das kann vorkommen, wenn ein Teilprogrammlauf, der vor dem Umschalten gestartet wurde, noch nicht abgeschlossen ist, und der zugehörige Prozess noch keine Meldung mit Meldungsziel SYSLOG geschrieben hat.

- Sie können mit *subopcode1*=KC_INFO abfragen, welche SYSLOG-Dateigenerationen bereits von allen UTM-Prozessen geschlossen worden sind. Das sind alle Dateigenerationen mit einer Generationsnummer kleiner *lowest_open_gen* (siehe *kc_syslog_str* im Abschnitt "KC_SYSLOG - System-Protokolldatei administrieren").

Wirkungsdauer / Transaktionssicherung / Cluster

Der Aufruf unterliegt nicht der Transaktionssicherung. Er wirkt unmittelbar und die Aktionen, die durch den Aufruf ausgelöst werden, sind bei der Rückkehr in das Teilprogramm bereits durchgeführt. Der Aufruf ist nicht rücksetzbar.

Das Ändern des Größenschwellwerts der SYSLOG-Datei wirkt bis zum Ende des Anwendungslaufes.

Liegt die Basis der SYSLOG-FGG innerhalb des gültigen Bereichs der SYSLOG-FGG (zwischen der ersten und der letzten Dateigeneration), dann protokolliert UTM im nächsten Anwendungslauf zunächst in die Basisdateigeneration. Liegt die Basis außerhalb des gültigen Bereichs, dann legt UTM beim nächsten Start für das Logging eine neue Dateigeneration an. Die Basis wird in der Datenstruktur *kc_syslog_str* im Feld mit *base_gen* angegeben.

In UTM-Cluster-Anwendungen gilt (Unix-, Linux- und Windows-Systeme):
Der Aufruf wirkt Cluster-global, d.h. die System-Protokolldatei SYSLOG wird für jede Knoten-Anwendung administriert. Die Größenüberwachung wirkt über den aktuellen Lauf der UTM-Cluster-Anwendung hinaus. Das Umschalten oder das Schreiben des Puffers wirkt nur für den aktuellen UTM-Cluster-Anwendungslauf, d.h. für alle Knoten-Anwendungen, die derzeit laufen.

KDCSLOG ("KDCSLOG - SYSLOG-Datei administrieren")

Versorgung der zu übergebenden Bereiche

Funktion des Aufrufs	Angabe im			
	Parameterbereich [1]	**Identifikati-onsbereich**	**Selektions-bereich**	**Datenbereich**
Über die SYSLOG informieren	*subopcode1*: KC_INFO *data_lth*: Länge des Datenbereichs für die Rückgabe von UTM	——	——	—— (Beim Aufruf müssen Sie den Zeiger auf einen Datenbereich für die Rückgaben von UTM (*kc_syslog_str)* übergeben.)
Schwellwert für die automatische Größenüberwachung setzen bzw. verändern	*subopcode1*: KC_CHANGE_SIZE *data_lth*: Länge der Daten im Datenbereich	——	——	Datenstruktur *kc_syslog_str* mit dem neuen Schwellwert
SYSLOG auf die nächste Dateigeneration der FGG umschalten	*subopcode1*: KC_SWITCH *data_lth*: 0	——	——	——
Schwellwert für die automatische Größenüberwachung verändern und SYSLOG auf die nächste Dateigeneration der FGG umschalten	*subopcode1*: KC_SWITCH_AND_CHANGE *data_lth*: Länge der Daten im Datenbereich	——	——	Datenstruktur *kc_syslog_str* mit dem neuen Schwellwert
Meldungspuffer in die SYSLOG schreiben	*subopcode1*: KC_WRITE_BUFFER *data_lth*: 0	——	——	——

[1] In allen Fällen muss im Parameterbereich der Operationscode KC_SYSLOG angegeben werden.

Versorgung der Parameter	
Parameterbereich	
Feldname	Inhalt
version	KC_ADMI_VERSION_1
retcode	KC_RC_NIL
version_data	KC_VERSION_DATA_11
opcode	KC_SYSLOG
subopcode1	KC_INFO / KC_CHANGE_SIZE / KC_SWITCH / KC_SWITCH_AND_CHANGE / KC_WRITE_BUFFER
id_lth	0
select_lth	0
data_lth	Länge der Datenstruktur / Länge des Datenbereichs / 0
Identifikationsbereich	
—	
Selektionsbereich	
—	
Datenbereich	
Datenstruktur kc_syslog_str / —	

KDCADMI-Aufruf
KDCADMI (&parameter_area, NULL, NULL, &data_area) KDCADMI (&parameter_area, NULL, NULL, NULL)

Rückgaben von UTM	
Parameterbereich	
Feldname	Inhalt
retcode	Returncodes
data_lth_ret	Länge der im Datenbereich gelieferten Daten
Datenbereich	
Datenstruktur kc_syslog_str	

subopcode1

Im Feld *subopcode1* müssen Sie angeben, welche Aktion UTM durchführen soll. Folgende Subopcodes können Sie angeben:

KC_WRITE_BUFFER

Alle an das Meldungsziel SYSLOG ausgegebenen Meldungen, die noch im UTM-internen Meldungspuffer zwischengespeichert sind, werden sofort in die aktuelle SYSLOG-Datei geschrieben. Ist der Puffer leer, hat der Aufruf keine Wirkung.

KC_INFO geben Sie an, wenn UTM Informationen über die SYSLOG-Datei bzw. SYSLOG-FGG zurückliefern soll. In diesem Fall müssen Sie im Feld *data_lth* die Länge des Datenbereichs angeben, den Sie UTM für die Übergabe der Informationen zur Verfügung stellen. Beim KDCADMI-Aufruf müssen Sie den Pointer auf diesen Datenbereich übergeben.

Folgende Werte für *subopcode1* dürfen Sie nur angeben, wenn die SYSLOG als FGG angelegt wurde.

KC_CHANGE_SIZE

geben Sie an, wenn Sie:

- den Schwellwert für die automatische Größenüberwachung ändern wollen. Den Schwellwert müssen Sie im Datenbereich übergeben.
- die automatische Größenüberwachung einschalten wollen. Dazu übergeben Sie im Datenbereich einen Schwellwert > '0'.
- die automatische Größenüberwachung ausschalten wollen. Dazu übergeben Sie im Datenbereich den Schwellwert '0'.

KC_SWITCH

geben Sie an, wenn UTM die SYSLOG-Datei auf die nächste Dateigeneration umschalten soll. Existiert diese Dateigeneration noch nicht, legt UTM sie an.

KC_SWITCH_AND_CHANGE

entspricht einer Zusammenfassung von KC_CHANGE_SIZE und KC_SWITCH. Mit KC_SWITCH_AND_CHANGE können Sie die SYSLOG auf die nächste Dateigeneration umschalten und gleichzeitig den Schwellwert für die automatische Größenüberwachung ändern. Dabei garantiert UTM, dass entweder beide Aktionen erfolgreich ausgeführt werden oder keine. D. h. nur wenn das Umschalten der SYSLOG erfolgreich war, stellt UTM den neuen Schwellwert ein.

Kann UTM nicht auf die folgende Dateigeneration umschalten, dann wird der Schwellwert nicht geändert. Die Größenüberwachung wird suspendiert und UTM ignoriert den neuen Schwellwert. Erst durch einen folgenden erfolgreichen Umschaltversuch (erneuter KC_SYSLOG-Aufruf) kann die Größenüberwachung wieder eingestellt werden. Wurde dabei kein neuer Schwellwert angegeben, übernimmt UTM den „alten“ Schwellwert.

data_lth

Im Feld *data_lth* geben Sie Folgendes an:

- bei *subopcode1*=KC_INFO:
  die Länge des Datenbereichs, in dem UTM die Informationen zurückliefern soll. Beim Aufruf von KDCADMI müssen Sie den Zeiger auf den Datenbereich an UTM übergeben.
- bei *subopcode1*=KC_CHANGE_SIZE oder KC_SWITCH_AND_CHANGE:
  die Länge der Daten im Datenbereich, die Sie an UTM übergeben. Im Datenbereich übergeben Sie die Datenstruktur *kc_syslog_str* mit dem neuen Schwellwert der Größenüberwachung.
- bei *subopcode1*=KC_SWITCH oder KC_WRITE_BUFFER: *data_lth*=0.
  Beim Aufruf von KDCADMI sollten Sie für *&data_area* den Nullpointer übergeben.

Datenbereich

Die Angaben, die Sie im Datenbereich machen müssen, sind abhängig von *subopcode1:*

- *subopcode1*=KC_WRITE_BUFFER oder KC_SWITCH:
  Sie dürfen im Datenbereich keine Daten an UTM übergeben.
- *subopcode1*=KC_INFO:
  Sie dürfen im Datenbereich keine Daten an UTM übergeben. Sie müssen UTM jedoch einen Datenbereich zur Verfügung stellen, in dem UTM die angeforderten Informationen zurückliefern kann.
- *subopcode1*=KC_CHANGE_SIZE oder KC_SWITCH_AND_CHANGE:
  Im Datenbereich müssen Sie die Datenstruktur *kc_syslog_str* mit dem neuen Schwellwert an UTM übergeben. Den Schwellwert geben Sie im Feld *size_control_utmpages* an. Er wird angegeben in Anzahl UTM-Seiten. Erlaubt sind Werte zwischen 0 und 2^{31}-1 (angegeben als char). Werte zwischen '1' und '99' werden von UTM jedoch automatisch durch '100' ersetzt. Mit *size_control_utmpages*='0' schalten Sie die automatische Größenüberwachung aus. Die restlichen Felder von *kc_syslog_str* müssen Sie mit binär null versorgen. *kc_syslog_str* ist im Abschnitt "KC_SYSLOG - System-Protokolldatei administrieren" beschrieben.

retcode

Im Feld *retcode* liefert UTM den Returncode des Aufrufs zurück. Neben den im Abschnitt „Returncodes" aufgelisteten Returncodes können zusätzlich folgende Returncodes auftreten.

Maincode = KC_MC_OK

Der Aufruf wurde fehlerfrei bearbeitet.

Subcodes:

KC_SC_MIN_SIZE

Bei *subopcode1*=KC_CHANGE_SIZE oder KC_SWITCH_AND_CHANGE:
Der Schwellwert der Größenüberwachung wurde zwar geändert, der in *size_control_utmpages* angegebene Wert war jedoch zu klein (< 100). Es wurde deshalb der minimale Schwellwert von 100 UTM-Seiten eingesetzt.

KC_SC_BUFFER_EMPTY

Bei *subopcode1*=KC_WRITE_BUFFER:
Der Meldungspuffer ist leer und wird daher nicht auf die SYSLOG geschrieben.

KC_SC_SWITCHED

Der Meldungspuffer konnte erst auf die SYSLOG geschrieben werden, nachdem auf eine neue Dateigeneration umgeschaltet wurde.

Maincode = KC_MC_REJECTED

Der Aufruf wurde von UTM abgewiesen.

Subcodes:

KC_SC_NO_FGG

Die gewünschte Aktion kann nicht durchgeführt werden, da SYSLOG nicht als FGG angelegt ist.

KC_SC_NO_INFO

Die Aktion kann nicht durchgeführt werden.

KC_SC_NO_GLOB_CHANG_POSSIBLE

Nur bei UTM-Cluster-Anwendungen:
Keine globalen Administrationsänderungen möglich, da die Generierung der Knoten-Anwendungen zur Zeit nicht konsistent ist.

Maincode = KC_MC_DATA_INVALID

Ein Feld der Datenstruktur im Datenbereich enthält einen ungültigen Wert.

Subcodes:

KC_SC_INVALID_MOD

Bei *subopcode1*=KC_CHANGE_SIZE oder KC_SWITCH_AND_CHANGE:
Der in *size_control_utmpages* angegebene Schwellwert der Größenüberwachung ist ungültig (zu groß, keine Zahl oder nicht abdruckbar). Der Schwellwert wurde daher nicht geändert.

KC_SC_DATA_MISSING

Bei *subopcode1*=KC_CHANGE_SIZE oder KC_SWITCH_AND_CHANGE:
In *size_control_utmpages* wurde kein Schwellwert für die Größenüberwachung angegeben. Er wurde deshalb nicht geändert und (bei KC_SWITCH_AND_CHANGE) die SYSLOG nicht umgeschaltet.

KC_SC_DATA_NOT_NULL

Bei *subopcode1*=KC_CHANGE_SIZE oder KC_SWITCH_AND_CHANGE:
In der Datenstruktur *kc_syslog_str* im Datenbereich wurde ein Feld, das nicht gesetzt werden darf, nicht mit binär null versorgt.

Maincode = KC_MC_RECBUF_FULL

Subcode:

KC_SC_NO_INFO

Der Puffer mit Wiederanlauf-Information ist voll (siehe openUTM-Handbuch „Anwendungen generieren“, KDCDEF-Steueranweisung MAX, Parameter RECBUF).

Maincode = KC_MC_REJECTED_CURR

Der Aufruf kann zur Zeit nicht bearbeitet werden.

Subcode:

KC_SC_INVDEF_RUNNING

Nur bei UTM-Cluster-Anwendungen:
Es läuft gerade ein inverser KDCDEF, d.h. der Auftrag kann z.Zt. nicht bearbeitet werden.

data_lth_ret

data_lth_ret enthält die Länge der Daten, die UTM im Datenbereich zurückliefert.

- bei *subopcode1*=KC_INFO liefert UTM die Informationen über die SYSLOG im Datenbereich (*kc_syslog_str*). Es ist *data_lth_ret* != 0.
- Ist die Länge in *data_lth_ret* kleiner als der bereitgestellte Datenbereich (*data_lth*), dann ist der Inhalt des Datenbereichs nur in der Länge *data_lth_ret* definiert.
- bei *subopcode1* != KC_INFO ist *data_lth_ret* = 0

Datenbereich

Im Fall *subopcode1*=KC_INFO liefert UTM im Datenbereich die Datenstruktur *kc_syslog_str* mit Informationen über die SYSLOG der Anwendung zurück. Die Datenstruktur hat folgende Felder:

struct kc_syslog_str
`char file_name[54];`
`char curr_size_utmpages[10];`
`char curr_size_kbyte[10];`
`char curr_size_percent[3];`
`char fgg;`
`char last_switch_ok;`
`char size_control_engaged;`
`char size_control_suspended;`
`char size_control_utmpages[10];`
`char size_control_kbyte[10];`
`char start_gen[4];`
`char curr_gen[4];`
`char lowest_open_gen[4];`
`char base_gen[4];`
`char first_valid_gen[4];`
`char last_valid_gen[4];`

Die Felder der Datenstruktur haben folgende Bedeutung:

file_name

Name der aktuellen SYSLOG-Datei bzw. der Dateigeneration, in die derzeit protokolliert wird.

curr_size_utmpages

enthält die momentane Größe der SYSLOG-Datei bzw. Dateigeneration, in die derzeit protokolliert wird. Die Größe wird angegeben in Anzahl der UTM-Seiten, die von der Datei bzw. Dateigeneration belegt sind.

curr_size_kbyte

enthält die momentane Größe der SYSLOG-Datei bzw. Dateigeneration, in die derzeit protokolliert wird. Die Größe wird angegeben in Kbyte.

curr_size_percent

Falls die automatische Größenüberwachung eingeschaltet ist, enthält *curr_size_percent* den Füllgrad der SYSLOG-Datei relativ zum eingestellten Größenschwellwert in Prozent. Ist die Größenüberwachung durch UTM suspendiert bzw. durch die Administration ausgeschaltet worden, dann kann der Füllgrad der SYSLOG-Datei auch größer als 100% sein. In diesem Fall liefert UTM in *curr_size_percent* Leerzeichen zurück. Ist keine Größenüberwachung definiert (weder per Generierung noch per Administration) dann belegt UTM *curr_size_percent* mit Leerzeichen.

fgg zeigt an, ob die SYSLOG als FGG oder als einfache Datei angelegt ist.

'Y' Die SYSLOG ist als FGG angelegt.

'N' Die SYSLOG ist als einfache Datei angelegt.

Alle folgenden Informationen sind nur relevant, wenn die SYSLOG als FGG angelegt ist. Ist die SYSLOG als einfache Datei angelegt, dann enthalten die folgenden Felder keine relevanten Informationen.

last_switch_ok

gibt an, ob der letzte Versuch von UTM, auf die nächste Dateigeneration umzuschalten, fehlerfrei abgelaufen ist. Die Aussage bezieht sich nur auf Umschaltversuche innerhalb des aktuellen Anwendungslaufs. Folgende Werte sind möglich:

'Y' Der letzte Umschaltversuch ist fehlerfrei abgelaufen.

'N' Beim letzten Umschaltversuch von UTM ist ein Fehler aufgetreten.
UTM konnte nicht auf die nächste Dateigeneration umschalten.

' ' (Leerzeichen) Im aktuellen Anwendungslauf gab es noch keinen Umschaltversuch oder die SYSLOG ist nicht als FGG angelegt.

size_control_engaged

gibt an, ob die automatische Größenüberwachung eingeschaltet ist. Folgende Werte sind möglich:

'Y' Größenüberwachung ist eingeschaltet

'N' Größenüberwachung ist ausgeschaltet

size_control_suspended

gibt an, ob die automatische Größenüberwachung durch UTM suspendiert wurde.

'Y' Der letzte Versuch, auf eine andere Dateigeneration umzuschalten, ist fehlgeschlagen. Aus diesem Grund ist die Größenüberwachung suspendiert.
UTM versucht nicht mehr auf die nächste Dateigeneration umzuschalten, auch wenn der eingestellte Größenschwellwert überschritten wird.

Maßnahme:
Sie können explizit versuchen, die SYSLOG umzuschalten. Verläuft das Umschalten fehlerfrei, so wird die Größenüberwachung durch UTM wieder aktiviert.

'N' Die Größenüberwachung ist nicht suspendiert.

size_control_utmpages

enthält den eingestellten Größenschwellwert der automatischen Größenüberwachung. Ausgegeben wird der Schwellwert in Anzahl UTM-Seiten.

size_control_utmpages='0' bedeutet, dass die Größenüberwachung ausgeschaltet ist.

In *size_control_utmpages* übergeben Sie bei *subopcode1*=KC_CHANGE_SIZE und KC_SWITCH_AND_CHANGE den neuen Größenschwellwert.

Minimalwert: '0'
Maximalwert: 2^{31} -1 (angegeben als char)

Geben Sie *size_control_utmpages*='0' an, dann wird die automatische Größenüberwachung ausgeschaltet. Werte zwischen '1' und '99' werden von UTM automatisch durch '100' ersetzt.

size_control_kbyte

enthält den eingestellten Größenschwellwert der automatischen Größenüberwachung. Ausgegeben wird der Schwellwert in KB. Bei sehr großen Schwellwerten wird der Kilobyte-Wert nicht angezeigt (z.B. bei 2^{31} KB).

size_control_kbyte=0 bedeutet, dass der Kilobyte-Wert nicht angezeigt werden kann, da er zu groß ist, oder dass die Größenüberwachung ausgeschaltet ist.

start_gen enthält die Generationsnummer der ersten SYSLOG-Dateigeneration, die UTM im aktuellen Anwendungslauf beschrieben hat.

curr_gen Generationsnummer der Dateigeneration, in die UTM gerade protokolliert.

lowest_open_gen

enthält die Generationsnummer der ältesten SYSLOG-Dateigeneration, die noch von einem Prozess der Anwendung offengehalten wird.

base_gen Generationsnummer der eingestellten Basis der SYSLOG-FGG.

first_valid_gen

Generationsnummer der ersten gültigen Dateigeneration der SYSLOG-FGG.

Auf BS2000-Systemen entspricht das der Angabe FIRST-GEN aus dem SHOW-FILE-ATTRIBUTES-Kommando.

last_valid_gen

Generationsnummer der letzten gültigen Dateigeneration der SYSLOG-FGG.

Auf BS2000-Systemen entspricht das der Angabe LAST-GEN aus dem Kommando SHOW-FILE-ATTRIBUTES.

11.2.16 KC_UPDATE_IPADDR - IP-Adressen aktualisieren

Mit KC_UPDATE_IPADDR können Sie im laufenden Betrieb der UTM-Anwendung die IP-Adressen, die in den Objekttabellen der Anwendung abgespeichert sind, mit den IP-Adressen abgleichen, die in der Hostnamen-Datenbank (hostname database) stehen. Die für Ihren Rechner relevante Hostnamen-Datenbank kann die `hosts`-Datei (Unix-, Linux- und Windows-Systeme), der DNS (Domain Name Service) oder auf BS2000-Systemen die Processor Table inklusive Socket Host Table sein.

Voraussetzung für einen Abgleich auf BS2000-Systemen ist, dass für den/die Partner der Protokolltyp SOCKET generiert ist.

UTM speichert die IP-Adressen folgender Kommunikationspartner in den Objekttabellen der UTM-Anwendung ab:

- Kommunikationspartner, die über die Socket-Schnittstelle (Transportprotokoll SOCKET) mit der UTM-Anwendung kommunizieren. Diese Kommunikationspartner sind als Clients vom Typ 'SOCKET' generiert (Partnertyp KC_PTERM).
- Nur auf Unix, Linunx- und Windows-Systemen: Kommunikationspartner, die über das Transportprotokoll RFC1006 mit der Anwendung kommunizieren. Das können Clients vom Typ='APPLI' oder 'UPIC-R' (KC_PTERM), LU6.1-Partner-Anwendungen (KC_CON) oder OSI TP-Partner-Anwendungen (KC_OSI_CON) sein.

Nähere Informationen zur Kommunikation über die Socket-Schnittstelle und zur Kommunikation über RFC1006 finden Sie im openUTM-Handbuch „Anwendungen generieren“.

Bei jedem Start der Anwendung liest UTM die IP-Adressen der Kommunikationspartner aus dem Name-Service und legt sie in den Objekttabellen ab.

Ändern sich während des Betriebs der Anwendung die IP-Adressen der betroffenen Kommunikationspartner, dann können Sie diesen Abgleich mit KC_UPDATE_IPADDR dynamisch anfordern.

Mit KC_UPDATE_IPADDR können Sie im Einzelnen:

- die IP-Adresse eines bestimmten Kommunikationspartners mit dem Eintrag im Name-Service abgleichen.
- die IP-Adressen aller betroffenen Kommunikationspartner mit den Adressen im Name-Service abgleichen.

Zur Kontrolle können Sie die IP-Adresse, die in der UTM-Anwendung für einen Kommunikationspartner gespeichert ist, mit KC_GET_OBJECT abfragen. UTM liefert die IP-Adresse im Feld *ip_addr* der Datenstruktur des Objekttyps zurück (*kc_con_str*, *kc_osi_con_str* oder *kc_pterm_str*).

Ablauf / Wirkungsdauer / Transaktionssicherung / Cluster

Der Aufruf unterliegt nicht der Transaktionssicherung. Er wirkt unmittelbar und die IP-Adressen sind bei der Rückkehr in das Teilprogramm bereits aktualisiert. Der Aufruf ist nicht rücksetzbar.

Die mit KC_UPDATE_IPADDR aktualisierten IP-Adressen bleiben in der UTM-Anwendung gespeichert bis zum Anwendungsende bzw. bis zum nächsten KC_UPDATE_IPADDR innerhalb des aktuellen Anwendungslaufs.

In UTM-Cluster-Anwendungen (Unix-, Linux- und Windows-Systeme) gilt:
Der Aufruf wirkt Cluster-global, d.h. der IP-Adressen-Abgleich wird auf jeder aktuell laufenden Knoten-Anwendung ausgeführt.

Versorgung der zu übergebenden Bereiche

Funktion des Aufrufs	Angabe im			
	Parameterbereich [1]	Identifikations-bereich	Selektions-bereich	Datenbereich
IP-Adresse eines Kommunikationspartners aktualisieren	*subopcode1*: KC_PARTNER *obj_type*: Partnertyp *obj_number*:1	Union *kc_id_area* mit dem Namen bzw. Namenstripel des Partners	——	Zeiger auf einen Datenbereich, in den UTM die Datenstruktur des Objekttyps mit der neuen IP-Adresse zurückliefert.
IP-Adressen aller betroffenen Kommunikationspartner mit der Datenbank für Hostnamen abgleichen	*subopcode1:* KC_ALL *obj_type*: KC_NO_TYPE *obj_number*:0	——	——	——

[1] In allen Fällen muss im Parameterbereich der Operationscode KC_UPDATE_IPADDR angegeben werden.

Versorgung der Parameter	
Parameterbereich	
Feldname	Inhalt
version	KC_ADMI_VERSION_1
retcode	KC_RC_NIL
version_data	KC_VERSION_DATA_11
opcode	KC_UPDATE_IPADDR
subopcode1	KC_PARTNER / KC_ALL
obj_type	KC_CON / KC_OSI_CON / KC_PTERM / KC_NO_TYPE
obj_number	1 / 0
id_lth	Länge des Partnernamens / 0
select_lth	0
data_lth	Länge des Datenbereichs / 0
Identifikationsbereich	
Partnername / —	

Selektionsbereich
—
Datenbereich
Datenstruktur des Objekttyps / —

KDCADMI-Aufruf
KDCADMI (&parameter_area, &identification_area, NULL, &data_area) oder KDCADMI (&parameter_area, NULL, NULL, NULL)

Rückgaben von UTM	
Parameterbereich	
Feldname	Feldinhalt
retcode	Returncode
data_lth_ret	Länge der im Datenbereich gelieferten Daten / 0
Datenbereich	
Datenstruktur des Objekttyps / —	

subopcode1

Im Feld *subopcode1* müssen Sie angeben:

KC_PARTNER

wenn UTM die IP-Adresse eines bestimmten Kommunikationspartners aktualisieren soll. Den Namen des Partners müssen Sie im Identifikationsbereich übergeben.

KC_ALL

wenn UTM die IP-Adressen aller Kommunikationspartner, die über das passende Protokoll mit der UTM-Anwendung kommunizieren, mit den Angaben in der Hostnamen-Datenbank abgleichen soll. Passende Protokolltypen sind:

- SOCKET
- RFC1006 (Unix-, Linux- und Windows-Systeme)

obj_type

Im Feld *obj_type* müssen Sie den Objekttyp des Kommunikationspartners angeben.

Bei *subopcode1*=KC_ALL müssen Sie *obj_type*=KC_NO_TYPE angeben.

Bei *subopcode1*=KC_PARTNER sind folgende Angaben möglich:

KC_PTERM

für Partner-Anwendungen, die als Clients von folgendem Typ konfiguriert sind.

- SOCKET (BS2000-Systeme)
- APPLI , UPIC-R oder SOCKET (Unix-, Linux- und Windows-Systeme)

KC_CON (Unix-, Linux- und Windows-Systeme)

für eine LU6.1-Partner-Anwendung.

KC_OSI_CON (Unix-, Linux- und Windows-Systeme)

für eine OSI TP-Partner-Anwendung.

obj_number

In *obj_number* müssen Sie die Anzahl der Objekte angeben, für die die IP-Adresse aktualisiert werden soll.

- bei *subopcode1*=KC_PARTNER müssen Sie *obj_number*=1 angeben
- bei *subopcode1*=KC_ALL müssen Sie *obj_number*=0 angeben. UTM aktualisiert dann die IP-Adressen aller entsprechend konfigurierten Kommunikationspartner.

id_lth

Welche Angabe Sie im Feld *id_lth* machen müssen, ist abhängig von der Angabe im Feld *subopcode1*:

- bei *subopcode1*=KC_PARTNER:
  müssen Sie in *id_lth* die Länge der Datenstruktur angeben, die Sie im Identifikationsbereich an UTM übergeben.
- bei *subopcode1*=KC_ALL:
  müssen Sie *id_lth*=0 setzen.

data_lth

Im Feld *data_lth* geben Sie die Länge des Datenbereichs an. Sie müssen Folgendes angeben:

- bei *subopcode1*=KC_PARTNER:
  Länge der Datenstruktur des Objekttyps in *obj_type*.
- bei *subopcode1*=KC_ALL:
  data_lth=0.

Identifikationsbereich

Wie Sie den Identifikationsbereich versorgen müssen, ist abhängig von *subopcode1*.

- bei *subopcode1*=KC_PARTNER:
  Im Identifikationsbereich müssen Sie die Union *kc_id_area* mit dem Namen des Kommunikationspartners übergeben. Die Angabe muss den Partner eindeutig identifizieren.

  Bei *obj_type*=KC_PTERM müssen Sie in der Struktur *kc_long_triple_str* der Union das Namenstripel aus Clientname (PTERM-Name), Prozessorname und BCAMAPPL-Name übergeben.

Bei *obj_type*=KC_CON auf Unix-, Linux- und Windows-Systemen müssen Sie in der Struktur *kc_long_triple_str* der Union das Namenstripel aus Anwendungsname, Prozessorname und BCAMAPPL-Name übergeben.

Bei *obj_type*=KC_OSI_CON auf Unix-, Linux- und Windows-Systemen müssen Sie im Feld *kc_name8* der Union den Namen der Verbindung zu der OSI TP-Partner-Anwendung angeben.

- bei *subopcode1*=KC_ALL müssen Sie den Nullpointer übergeben.

Datenbereich

Wie Sie den Datenbereich versorgen müssen, ist abhängig von *subopcode1*:

- bei *subopcode1*=KC_PARTNER geben Sie die Datenstruktur des Objekttyps an (*kc_con_str*, *kc_osi_con_str* oder *kc_pterm_str*).
- bei *subopcode1*=KC_ALL müssen Sie den Nullpointer übergeben.

retcode

Im Feld *retcode* liefert UTM den Returncode des Aufrufs zurück. Neben den im Abschnitt „Returncodes" aufgelisteten Returncodes können zusätzlich folgende Returncodes auftreten

Maincode = KC_MC_REJECTED Der Aufruf wurde von UTM abgewiesen. **Subcodes:**
KC_SC_TPROT_NOT_ALLOWED (nur auf Unix-, Linux- und Windows-Systemen) Transportprotokoll wird nicht unterstützt, d.h. in der Anwendung sind keine Kommunikationspartner für die Kommunikation über SOCKET generiert. Dieser Returncode kann auch auftreten, wenn in der Anwendung zwar ein Kommunikationspartner für die Kommunikation über SOCKET generiert ist, z.B. ein BCAMAPPL, aber KC_PARTNER mit Objekttyp KC_CON oder KC_OSI_CON angegeben wird. Auf BS2000-Systemen ist bei KC_PARTNER nur die Angabe von Objekt KC_PTERM möglich. Dieser Returncode wird auch zurückgeliefert, wenn in der Anwendung nicht mindestens ein Kommunikationspartner und das zugehörige BCAMAPPL mit T-PROT=SOCKET generiert sind.
KC_SC_SOCKET_ERROR Ein Abgleich der IP-Adresse(n) konnte aufgrund eines Fehlers an der Kommunikations-Schnittstelle (Socket-Aufruf) nicht vorgenommen werden.
KC_SC_INVALID_NAME Der im Identifikationsbereich angegebene Kommunikationspartner existiert nicht oder er kommuniziert nicht über das passende Transportprotokoll mit der UTM-Anwendung.
KC_SC_NO_IPADDR_FOUND *subopcode1*=KC_PARTNER: Zu dem angegebenen Kommunikationspartner wurde im Name-Service keine IP-Adresse gefunden. *subopcode1*=KC_ALL:

UTM konnte im Name-Service zu keinem Kommunikationspartner des angegebenen Objekttyps eine IP-Adresse finden.

KC_SC_AT_LEAST_ONE_OBJ_FAILED

Ein Abgleich der IP-Adressen mit *subopcode1*=KC_ALL ist erfolgt. Es ist aber bei mindestens einem Objekt ein Fehler aufgetreten.
Mögliche Ursachen können die bei den vorhergehenden Returncodes beschriebenen Fehler sein.
In der Meldung K154, die standardmäßig auf SYSLOG und SYSOUT erfolgt, finden Sie die Information, bei welchem Partner/welchen Partnern ein Fehler aufgetreten ist.

KC_SC_NO_GLOB_CHANG_POSSIBLE

Nur bei UTM-Cluster-Anwendungen:
Keine globale Administration erlaubt, da die Generierung der Knoten-Anwendungen zur Zeit nicht konsistent ist.

Maincode = KC_MC_RECBUF_FULL

Subcode:

KC_SC_NO_INFO

Der Puffer mit Wiederanlauf-Information ist voll (siehe openUTM-Handbuch „Anwendungen generieren “, KDCDEF-Steueranweisung MAX, Parameter RECBUF).

Maincode = KC_MC_REJECTED_CURR

Der Aufruf kann zur Zeit nicht bearbeitet werden.

Subcode:

KC_SC_INVDEF_RUNNING

Nur bei UTM-Cluster-Anwendungen:
Es läuft gerade ein inverser KDCDEF, d.h. der Auftrag kann z.Zt. nicht bearbeitet werden.

data_lth_ret

data_lth_ret enthält die Länge der Daten, die UTM im Datenbereich zurückliefert.

- bei *subopcode1*=KC_PARTNER: Länge der Daten, die UTM im Datenbereich zurückliefert
- bei *subopcode1*=KC_ALL: *data_lth_ret*=0

Datenbereich

Im Fall *subopcode1*=KC_PARTNER liefert UTM im Datenbereich die Datenstruktur des Objekttyps (*kc_con_str* , *kc_osi_con_str* oder *kc_pterm_str)* mit folgenden Informationen zurück .

- Ist die neue IP-Adresse des Kommunikationspartners eine IPv4-Adresse, steht sie im Feld *ip_addr* der Datenstruktur in der Länge 15. Im Feld *ip_v* steht V4.
- Ist die neue IP-Adresse des Kommunikationspartners eine IPv6-Adresse, steht sie im Feld *ip_addr_v6* der Datenstruktur in der Länge 39. Im Feld *ip_v* steht V6.

- Die anderen Felder der Datenstruktur sind nicht versorgt.

11.2.17 KC_USLOG - Benutzer-Protokolldatei administrieren

Die Benutzer-Protokolldatei wird als Dateigenerationsverzeichnis USLOG geführt. Mit KC_USLOG können Sie die aktuelle Benutzer-Protokolldatei (Dateigeneration von USLOG) schließen und gleichzeitig eine neue Benutzer-Protokolldatei eröffnen. Das ist die Dateigeneration mit der nächstfolgenden Generationsnummer. Die geschlossene Protokolldatei ist dann beliebig verwendbar.

Umschalten bei doppelter USLOG

Wird die Benutzer-Protokolldatei Ihrer Anwendung doppelt geführt (siehe openUTM-Handbuch „Anwendungen generieren“), dann wirkt der KC_USLOG-Aufruf auf beide Dateien.

Wirkungsdauer der Änderung / Cluster

Die erfolgreiche Bearbeitung des Aufrufs bedeutet, dass das Umschalten auf die nächste Dateigeneration erfolgreich verlaufen ist. UTM schreibt alle nach dem Umschaltzeitpunkt erzeugten LPUT-Nachrichten in die neue Protokolldatei. Nach dem Umschalten schreibt UTM die LPUT-Nachrichten solange in die neue(n) USLOG-Dateigeneration(en), bis Sie erneut auf die folgende Dateigeneration umschalten.

In UTM-Cluster-Anwendungen (Unix-, Linux- und Windows-Systeme) gilt:
Der Aufruf wirkt Cluster-global, d.h. das Schließen der aktuellen Benutzer-Protokolldatei und das Öffnen einer neuen Benutzer-Protokolldatei wird in jeder aktuell laufenden Knoten-Anwendung ausgeführt.

KDCLOG ("KDCLOG - Benutzer-Protokolldatei umschalten")

Versorgung der zu übergebenden Bereiche

Funktion des Aufrufs	Angabe im			
	Parameterbereich [1]	Identifikati-onsbereich	Selektions-bereich	Datenbereich
Benutzer-Protokolldatei auf die nächste Dateigeneration der FGG umschalten	*subopcode1*: KC_SWITCH *data_lth*: 0	——	——	——

[1] Im Parameterbereich muss der Operationscode KC_USLOG angegeben werden.

Versorgung der Parameter	
Parameterbereich	
Feldname	Inhalt
version	KC_ADMI_VERSION_1
retcode	KC_RC_NIL
version_data	KC_VERSION_DATA_11
opcode	KC_USLOG
subopcode1	KC_SWITCH
id_lth	0
select_lth	0
data_lth	0
Identifikationsbereich	
—	
Selektionsbereich	
—	
Datenbereich	
—	

KDCADMI-Aufruf
KDCADMI (&parameter_area, NULL, NULL, NULL)

Rückgaben von UTM	
Parameterbereich	
Feldname	Inhalt
retcode	Returncodes

retcode

Im Feld *retcode* liefert UTM den Returncode des Aufrufs zurück. Neben den im Abschnitt „Returncodes" aufgelisteten Returncodes können zusätzlich folgende Returncodes auftreten:

Maincode = KC_MC_REJECTED_CURR Der Aufruf kann zur Zeit nicht bearbeitet werden. **Subcode:**
KC_SC_LWRT_IN_PROGRESS Die USLOG kann zu diesem Zeitpunkt nicht auf die nächste Dateigeneration umgeschaltet werden, da UTM gerade Daten in die USLOG schreibt.
KC_SC_INVDEF_RUNNING Nur bei UTM-Cluster-Anwendungen: Es läuft gerade ein inverser KDCDEF, d.h. der Auftrag kann z.Zt. nicht bearbeitet werden.

Maincode = KC_MC_REJECTED Der Aufruf wurde von UTM abgewiesen. **Subcode:**
KC_SC_FILE_ERROR Das Umschalten der USLOG auf die nächste Dateigeneration ist wegen eines DMS-Fehlers nicht möglich.
KC_SC_NO_GLOB_CHANG_POSSIBLE Nur bei UTM-Cluster-Anwendungen: Keine globale Administration erlaubt, da die Generierung der Knoten-Anwendungen zur Zeit nicht konsistent ist.

Maincode = KC_MC_RECBUF_FULL **Subcode:**
KC_SC_NO_INFO Der Puffer mit Wiederanlauf-Information ist voll (siehe openUTM-Handbuch „Anwendungen generieren “, KDCDEF-Steueranweisung MAX, Parameter RECBUF).

11.3 Datenstrukturen zur Informationsübergabe

In diesem Abschnitt werden die Datenstrukturen beschrieben, die Sie bei den Aufrufen von KC_GET_OBJECT, KC_MODIFY_OBJECT oder KC_CREATE_OBJECT über den Datenbereich legen müssen.

- Bei KC_GET_OBJECT liefert UTM die angeforderten Objekteigenschaften, Anwendungsparameter und Statistikdaten im Format dieser Datenstrukturen zurück.
- Bei der Modifikation von Objekteigenschaften und Anwendungsparametern (KC_MODIFY_OBJECT) sowie beim dynamischen Eintragen neuer Objekte in die Konfiguration (KC_CREATE_OBJECT) werden die Daten in diesem Format an UTM übergeben.

Die Datenstrukturen sind in der Include-Datei *kcadminc.h* definiert.

In den folgenden zwei Abschnitten werden die Datenstrukturen und die Bedeutung ihrer Elemente beschrieben.
Der Abschnitt „Datenstrukturen zur Beschreibung der Objekteigenschaften" beschreibt die Strukturen, in denen Informationen über Objekte der Anwendung übergeben werden.
Der Abschnitt „Datenstrukturen zur Beschreibung der Anwendungsparameter" beschreibt die Strukturen, in denen Anwendungsparameter übergeben werden.

Zusätzlich zu den in diesen Abschnitten beschriebenen gibt es weitere Datenstrukturen, die zu keinem Objekt- oder Parametertyp gehören. Sie werden bei bestimmten Aufrufen benötigt, um Daten mit UTM auszutauschen. Diese Datenstrukturen sind bei den entsprechenden Operationscodes beschrieben.

Die Namen dieser Datenstrukturen setzen sich wie folgt zusammen: *operationscode_str*.

Zu dieser Gruppe gehören folgende Datenstrukturen:

- *kc_change_application_str* benötigen Sie zur Übergabe von Daten beim Programmaustausch mit KC_CHANGE_APPLICATION ("KC_CHANGE_APPLICATION - Anwendungsprogramm austauschen").
- *kc_create_statements_str* benötigen Sie, um beim Anfordern des inversen KDCDEF-Laufs mit KC_CREATE_STATEMENTS Daten an UTM zu übergeben ("KC_CREATE_STATEMENTS - KDCDEF-Steueranweisungen erzeugen (inverser KDCDEF)").
- *kc_encrypt_advanced_str* bzw. *kc_encrypt_str* benötigen Sie für das Auslesen des Public-Schlüssels eines RSA-Schlüsselpaares mit KC_ENCRYPT ("KC_ENCRYPT - RSA-Schlüsselpaar erzeugen, löschen, auslesen").
- *kc_shutdown_str* benötigen Sie, um beim Anfordern eines Shutdown mit KC_SHUTDOWN Daten an UTM zu übergeben ("KC_SHUTDOWN - Anwendungslauf beenden").
- *kc_syslog_str* wird benötigt bei der Administration der SYSLOG-Datei mit KC_SYSLOG ("KC_SYSLOG - System-Protokolldatei administrieren").
- *kc_online_import_str* benötigen Sie, um Anwendungsdaten mit KC_ONLINE_IMPORT online zu importieren.
- *kc_lock_mgmt_str* benötigen Sie, um Sperren in UTM-Cluster-Anwendungen mit KC_LOCK_MGMT aufzuheben.

Allgemeines zum Aufbau der Datenstrukturen

Die Felder der Datenstrukturen bestehen nur aus dem Datentyp „char“. Die eckigen Klammern nach den Feldnamen beinhalten die Länge des Feldes. Fehlt die eckige Klammer, dann ist das Feld 1 Byte lang.

Beim Austausch von Daten zwischen UTM und einem Administrations-Teilprogramm ist Folgendes zu beachten:

- Namen und Schlüsselwörter werden linksbündig abgelegt und rechts mit Leerzeichen aufgefüllt.
  Die Übergabe der Daten an UTM muss, außer bei Objektnamen, in Großbuchstaben erfolgen.

Beispiel
Das Feld *ptype* (*kc_pterm_str*) ist 8 Byte lang. *ptype*=APPLI wird wie folgt abgelegt: 'APPLI*bbb*' (*b* = blank).

- Numerische Angaben werden von UTM rechtsbündig abgelegt und mit führenden Leerzeichen geliefert. Bei der Datenübergabe von einem Administrationsprogramm an UTM werden links- und rechtsbündige Angaben akzeptiert. Rechtsbündige Angaben werden mit führenden Leerzeichen oder Nullen akzeptiert. Linksbündige Angaben können auch mit dem Null-Byte (\0) abgeschlossen oder mit Leerzeichen aufgefüllt werden.

  Beispiel
  Das Feld *conn_users* (*kc_max_par_str*) ist 10 Byte lang. *conn_users*=155 wird von UTM wie folgt abgelegt: '*bbbbbbb*155' (*b* = blank).
- Bei der Übergabe von Daten an UTM sind Felder der Datenstrukturen, in denen kein Wert angegeben wird, mit binär null zu versorgen.

Beschreibungsformat

Die Datenstrukturen von *kcadminc.h* werden in Tabellen dargestellt. Die Tabellen sind wie folgt aufgebaut:

mod	Datenstructur kc_..._str	Seite
1.	**2.**	**3.**

1. In der 1. Spalte (grau gerastert) wird angegeben, welche Parameter, d.h. Feldinhalte, Sie mit KC_MODIFY_OBJECT modifizieren können. Fehlt die Spalte „mod“, dann können keine Parameter modifiziert werden.

   Die in der 1. Spalte verwendeten Abkürzungen haben die folgende Bedeutung:

-	Der Parameter kann nicht modifiziert werden.
x(*y*)	Der Wert des Parameters kann modifiziert werden.
	Der Wert in der Klammer (*y*) gibt Auskunft über Wirksamkeit und Lebensdauer der Änderung. *y* kann die Werte IR/GIR, ID/GID , PR/GPR, PD/GPD, P/GP, A/GA annehmen. Die Bedeutung der Abkürzungen finden Sie im Abschnitt "KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern".

2. Die 2. Spalte enthält die Felder der Datenstruktur, wie sie in *kcadminc.h* definiert sind.
3. Die 3. Spalte wird nur bei der Darstellung sehr großer Datenstrukturen verwendet. In dieser Spalte wird dann die Seite angegeben, auf der Sie die Beschreibung zu dem jeweiligen Datenstrukturfeld finden.

Im Anschluss an die jeweilige Tabelle wird die Bedeutung der Feldinhalte beschrieben.

11.3.1 Datenstrukturen zur Beschreibung der Objekteigenschaften

Im Folgenden werden alle Datenstrukturen beschrieben, die für die Übergabe von Objekteigenschaften zur Verfügung stehen. Für jeden einzelnen Objekttyp steht eine eigene Datenstruktur zur Verfügung. Diese Datenstrukturen finden Sie in der Include-Datei *kcadminc.h*. Der Name der jeweiligen Datenstruktur setzt sich zusammen aus dem Namen des Objekttyps und dem Suffix „*_str*“. Die Beschreibung erfolgt in alphabetisch aufsteigender Reihenfolge der Datenstrukturnamen.

> ¡ Datenstrukturen können am Ende *filler*-Felder enthalten. Diese sind hier nicht aufgeführt.

11.3.1.1 kc_abstract_syntax_str - Abstrakte Syntax für die Kommunikation über OSI TP

Für den Objekttyp KC_ABSTRACT_SYNTAX ist die Datenstruktur *kc_abstract_syntax_str* definiert. Bei KC_GET_OBJECT liefert UTM in *kc_abstract_syntax_str* für eine Abstrakte Syntax den lokalen Namen, den Object-Identifier und den Namen der zugewiesenen Transfersyntax zurück.

Die Abstrakte Syntax gibt bei der Kommunikation über OSI TP an, wie die Benutzerdaten vor dem Übertragen zum Kommunikationspartner codiert werden. Beide Kommunikationspartner müssen dieselbe Abstrakte Syntax verwenden.

Datenstruktur kc_abstract_syntax_str
`char abstract_syntax_name [8];`
`char object_id[10][8];`
`char transfer_syntax[8];`

Die Felder der Datenstruktur haben die folgende Bedeutung:

abstract_syntax_name

enthält den lokalen Namen der Abstrakten Syntax.

Der lokale Name muss beim MGET/MPUT bzw. FGET/FPUT angegeben werden, wenn Daten dieser Abstrakten Syntax gesendet oder empfangen werden sollen.

object_id

enthält den Object-Identifier der Abstrakten Syntax.

Der Object-Identifier besteht aus mindestens 2, maximal aber aus 10 Komponenten. Die einzelnen Komponenten sind positive ganze Zahlen im Bereich von 0 bis 67 108 863.

UTM liefert pro Komponente des Object-Identifiers ein Feldelement zurück, d.h. die Anzahl der belegten Feldelemente in *object_id* entspricht der Anzahl der Komponenten. Die restlichen Feldelemente sind mit binär null versorgt.

Näheres zum Object-Identifier finden Sie im openUTM-Handbuch „Anwendungen generieren“.

transfer_syntax

enthält den lokalen Namen der Transfersyntax, die der Abstrakten Syntax zugewiesen ist.

11.3.1.2 kc_access_point_str - OSI TP-Zugriffspunkt

Für den Objekttyp KC_ACCESS_POINT ist die Datenstruktur *kc_access_point_str* definiert. In *kc_access_point_str* liefert UTM bei KC_GET_OBJECT Namen und Adresse eines lokalen OSI TP-Zugriffspunkts zurück.

Ein lokaler OSI TP-Zugriffspunkt wird mit der KDCDEF-Steueranweisung ACCESS-POINT statisch generiert.

Datenstruktur kc_access_point_str
`char ap_name[8];`
`char application_entity_qualifier[8];`
`union kc_selector presentation_selector;`
`union kc_selector session_selector;`
`char presentation_selector_type;`
`char presentation_selector_lth[2];`
`char presentation_selector_code;`
`char session_selector_type;`
`char session_selector_lth[2];`
`char session_selector_code;`
`char transport_selector[8];`
`char listener_id[5];` (nur auf Unix-, Linux- und Windows-Systemen)
`char listener_port[5];` (nur auf Unix-, Linux- und Windows-Systemen)
`char t_prot[6];` (nur auf Unix-, Linux- und Windows-Systemen)
`char tsel_format;` (nur auf Unix-, Linux- und Windows-Systemen)

Die Felder der Datenstruktur haben die folgende Bedeutung:

ap_name

Name des OSI TP-Zugriffspunktes. Innerhalb der lokalen UTM-Anwendung wird der OSI TP-Zugriffspunkt durch diesen Namen eindeutig identifiziert.

application_entity_qualifier

Application Entity Qualifier (AEQ) des Zugriffspunktes. Der AEQ wird bei der Kommunikation mit einigen heterogenen Kommunikationspartnern für die Adressierung benötigt. Diese Kommunikationspartner adressieren den Zugriffspunkt über den Application Process Titel (APT) der lokalen Anwendung und den AEQ des Zugriffspunktes.

Der AEQ ist eine positive ganze Zahl zwischen 1 und 67108863 (= 2^{26}-1). Näheres zum AEQ finden Sie im openUTM-Handbuch „Anwendungen generieren“.

application_entity_qualifier='0' bedeutet, dass für den Zugriffspunkt kein AEQ definiert ist.

presentation_selector

enthält den Presentation-Selektor der Adresse des OSI TP-Zugriffspunktes.

presentation_selector ist ein Feld vom Typ *kc_selector*:

union kc_selector
`char x[32];`
`char c[16];`

UTM liefert den Presentation-Selektor i.A. als Character-String (*c*) in Maschinen-spezifischer Codierung (*presentation_selector_code*='S') zurück. Der Character-String ist maximal 16 Zeichen lang. Das Feld *presentation_selector* ist ab der in *presentation_selector_lth* angegebenen Länge mit Leerzeichen aufgefüllt.

In Ausnahmefällen wird der Presentation-Selektor als hexadezimale Zeichenfolge (*x*) zurückgeliefert. Jedes Halb-Byte wird dabei als ein Zeichen dargestellt, z.B. die Hexadezimalzahl A2 wird als String 'A2 ' (2 Zeichen) zurückgeliefert. Ist der Presentation-Selektor eine Hexadezimalzahl, dann liefert UTM bis zu 32 Byte zurück.

Wie der Inhalt von *presentation_selector* zu interpretieren ist, entnehmen Sie dem Feld *presentation_selector_type.*

Enthält die Adresse des Zugriffspunktes keinen Presentation-Selektor, dann ist das Feld *presentation_selector* mit Leerzeichen belegt. In diesem Fall ist *presentation_selector_type*='N' und *presentation_selector_lth*='0'.

session_selector

enthält den Session-Selektor der Adresse des OSI TP-Zugriffspunktes.

session_selector ist eine Union vom Typ *kc_selector (*siehe *presentation_selector*).

UTM liefert den Session-Selektor i.A. als Character-String (*c*) in Maschinen-spezifischer Codierung (*session_selector_code*='S') zurück. Der Character-String ist maximal 16 Zeichen lang. Das Feld *session_selector* ist ab der in *session_selector_lth* angegebenen Länge mit Leerzeichen aufgefüllt.

In Ausnahmefällen wird der Session-Selektor als hexadezimale Zeichenfolge (*x*) zurückgeliefert. Jedes Halb-Byte wird dabei als ein Zeichen dargestellt. Ist der Session-Selektor eine Hexadezimalzahl, dann liefert UTM in *session_selector* bis zu 32 Byte zurück.

Wie der Inhalt von *session_selector* zu interpretieren ist, entnehmen Sie dem Feld *session_selector_type.*

Enthält die Adresse des Zugriffspunktes keinen Session-Selektor, dann ist das Feld *session_selector* mit Leerzeichen belegt. In diesem Fall ist *session_selector_type*='N' und *session_selector_lth*='0'.

presentation_selector_type

gibt an, ob die Adresse des Zugriffspunktes einen Presentation-Selektor enthält und wie die Rückgabe in *presentation_selector* zu interpretieren ist.

'N' steht für *NONE. Die Adresse des Zugriffspunktes enthält keinen Presentation-Selektor, *presentation_selector* ist mit Leerzeichen belegt und *presenta tion_selector_lth*='0'.

'C' Die Angabe des Presentation-Selektor in *presentation_selector* ist als Character-String zu interpretieren. Es sind maximal die ersten 16 Byte von *presentation_selector* belegt.

'X' Der Presentation-Selektor in *presentation_selector* ist eine Hexadezimalzahl.

presentation_selector_lth

enthält die Länge des Presentation-Selektors (*presentation_selector)* in Byte. Ist *presentation_selector_lth* ='0', dann enthält die Adresse des OSI TP-Zugriffspunktes keine Presentation-Komponente (*presentation_selector* enthält Leerzeichen).
Sonst liegt der Wert von *presentation_selector_lth* zwischen '1' und '16'.

Ist *presentation_selector_type*='X', dann ist die Länge des in *presentation_selector* angegebenen Strings: 2 * *presentation_selector_lth* Byte.

Beispiel
Der Presentation-Selektor des Zugriffspunktes ist X'A2B019CE'. *presentation_selector* enthält dann den String 'A2B019CE', *presentation_selector_type*='X' und *presentation_selector_lth*=' 4' (vier Hexadezimalziffern).

presentation_selector_code

gibt an, wie der Presentation-Selektor in *presentation_selector* codiert ist.

UTM liefert 'S' zurück, wenn der Presentation-Selektor als Character-String zurückgeliefert wird (*presentation_selector_type*='C').

'S' bedeutet: Maschinen-spezifische Codierung (Standardcodierung, EBCDIC auf BS2000-Systemen, ASCII auf Unix-, Linux- und Windows-Systemen).

Ist *presentation_selector_type*='X' oder 'N', dann liefert UTM im Feld *presentation_selector_code* ein Leerzeichen zurück.

session_selector_type

gibt an, ob die Adresse des Zugriffspunktes einen Session-Selektor enthält und wie die Rückgabe in *session_selector* zu interpretieren ist.

'N' steht für *NONE. Die Adresse des Zugriffspunktes enthält keinen Session-Selektor. Das Feld *session_selector* enthält Leerzeichen und *session_selec tor_lth*='0'.

'C' Die Angabe des Session-Selektors in *session_selector* ist als Character-String zu interpretieren. Es sind maximal die ersten 16 Byte von *session_selector* belegt.

'X' Der Session-Selektor in *session_selector* ist eine Hexadezimalzahl.

session_selector_lth

enthält die Länge des Session-Selektors (*session_selector)* in Byte.

Ist *session_selector_lth*='0', dann enthält die Adresse des Zugriffspunktes keine Session-Komponente (*session_selector* enthält Leerzeichen). Sonst liegt der Wert von *session_selector_lth* zwischen '1' und '16'.

Ist *session_selector_type*='X', dann ist die Länge des in *session_selector* enthaltenen Strings: 2 * *session_selector_lth* Byte.

session_selector_code

gibt an, wie der Session-Selektor in *session_selector* codiert ist.

UTM liefert 'S' zurück, wenn der Session-Selektor als Character-String zurückgeliefert wird (*session_selector_type*='C').

'S' bedeutet: Maschinen-spezifische Codierung (Standardcodierung, EBCDIC auf BS2000-Systemen, ASCII auf Unix-, Linux- und Windows-Systemen).

Ist *session_selector_type*='X' oder 'N', dann liefert UTM im Feld *session_selector_code* ein Leerzeichen zurück.

transport_selector

enthält den Transport-Selektor der Adresse des OSI TP-Zugriffspunktes. *transport_selector* enthält immer einen gültigen Wert, da jedem Zugriffspunkt bei der KDCDEF-Generierung ein Transport-Selektor zugeordnet werden muss. Der Transport-Selektor ist immer als Character-String zu interpretieren, er besteht aus 1 bis 8 abdruckbaren Zeichen.

Auf BS2000-Systemen ist der Wert von *transport_selector* ein lokaler BCAM-Anwendungsname.

listener_id (nur auf Unix-, Linux- und Windows-Systemen)

enthält die Listener-Id des Zugriffspunktes. Die Listener-Id ist eine positive ganze Zahl zwischen 0 und 32767.

Mit der Listener-Id wird festgelegt, welche Verbindungen von demselben Netzprozess verwaltet werden sollen. Alle Verbindungen, die über Zugriffspunkte und BCAMAPPL-Namen mit derselben Listener-Id aufgebaut werden, werden von einem Netzprozess verwaltet.

Ausnahmen sind BCAMAPPL-Namen für die Kommunikation über die Socket-Schnittstelle (SOCKET). Diese bilden einen eigenen Nummernkreis. Es werden also keine Access-Points mit solchen BCAMAPPL-Namen in einem Netzprozess zusammengefasst, auch wenn die *listener-id* gleich ist.

Die folgenden Felder haben nur für Zugriffspunkte einer UTM-Anwendung auf Unix-, Linux- und Windows-Systemen eine Bedeutung. Die Felder enthalten die Komponenten der Adresse des Zugriffspunktes innerhalb des lokalen Systems. Siehe dazu auch openUTM-Handbuch „Anwendungen generieren“.

listener_port

enthält die Portnummer des Zugriffspunktes für den Aufbau von TCP/IP-Verbindungen. Es wird die Portnummer angegeben, die bei der KDCDEF-Generierung angegeben wurde.

listener_port='0' bedeutet, dass bei der Generierung keine Portnummer angegeben wurde.

t_prot

enthält das Adressformat, das dem Zugriffspunkt bei der KDCDEF-Generierung zugeordnet wurde.

Die Adressformate werden wie folgt angegeben:

'R' RFC1006, ISO-Transportprotokoll Klasse 0 über TCP/IP und Konvergenzprotokoll RFC1006.

Enthält *t_prot* nur Leerzeichen, dann wurde bei der KDCDEF-Generierung kein Adressformat definiert.

tsel_format

gibt das Format der T-Selektoren in der Adresse des Zugriffspunkts an (Formatindikator).

'T' TRANSDATA-Format

'E' EBCDIC-Zeichenformat

'A' ASCII-Zeichenformat

Enthält *tsel_format* ein Leerzeichen, dann wurde bei der KDCDEF-Generierung kein Formatindikator definiert.

Zur Bedeutung der Adressformate siehe „Dokumentation zu PCMX“ im Abschnitt "openUTM-Dokumentation".

11.3.1.3 kc_application_context_str - Application Context für die Kommunikation über OSI TP

Für den Objekttyp KC_APPLICATION_CONTEXT ist die Datenstruktur *kc_application_context_str* definiert. In *kc_application_context_str* liefert UTM bei KC_GET_OBJECT den lokalen Namen und die Eigenschaften eines Application Context zurück.

Der Application Context legt die Regeln der Datenübertragung zwischen den Kommunikationspartnern fest. Er gibt an, wie die Benutzerdaten für die Übertragung codiert werden (Abstrakte Syntax) und in welcher Form die Daten übertragen werden (Transfersyntax).

Der Application Context muss mit dem Partner abgestimmt werden. Näheres zum Application Context finden Sie im openUTM-Handbuch „Anwendungen generieren“.

Datenstruktur kc_application_context_str
`char application_context_name [8];`
`char object_id[10][8];`
`char abstract_syntax[9][8];`

Die Felder der Datenstruktur haben die folgende Bedeutung:

application_context_name

enthält den Namen, der lokal für den Application Context generiert wurde.

object_id

enthält den Object-Identifier des Application Context.

Der Object-Identifier besteht aus mindestens 2, maximal aber aus 10 Komponenten. Die einzelnen Komponenten sind positive ganze Zahlen im Bereich von 0 bis 67108863.

UTM liefert pro Komponente des Object-Identifiers ein Feldelement zurück, d.h. die Anzahl der belegten Feldelemente in *object_id* entspricht der Anzahl der Komponenten. Die restlichen Feldelemente sind mit binär null versorgt.

Näheres zum Object-Identifier finden Sie im openUTM-Handbuch „Anwendungen generieren“.

abstract_syntax

enthält die lokalen Namen der Abstrakten Syntaxen, die dem Application Context zugeordnet sind. Einem Application Context können bis zu 9 Abstrakte Syntaxen zugeordnet sein. Pro Abstrakter Syntax liefert UTM ein Feldelement zurück, d.h. die Anzahl der belegten Feldelemente in *abstract_syntax* entspricht der Anzahl der Abstrakten Syntaxen, die dem Application Context zugeordnet sind. Die restlichen Feldelemente sind mit binär null versorgt.

Jedem Application Context ist mindestens eine Abstrakte Syntax zugewiesen.

11.3.1.4 kc_bcamappl_str - Namen und Adressen der lokalen Anwendung

Für den Objekttyp KC_BCAMAPPL ist die Datenstruktur *kc_bcamappl_str* definiert. In *kc_bcamappl_str* liefert UTM bei KC_GET_OBJECT die Namen und Eigenschaften der lokalen Anwendung zurück.

UTM informiert über die Eigenschaften der lokalen Anwendung, die dem in MAX APPLINAME definierten Namen der Anwendung bzw. den BCAMAPPL-Namen der Anwendung zugeordnet sind. BCAMAPPL-Namen sind zusätzliche Namen der Anwendung für die verteilte Verarbeitung mit LU6.1 und für den Anschluss von Clients. Sie werden mit der KDCDEF-Anweisung BCAMAPPL generiert. Über die Namen der Anwendung werden die Verbindungen zwischen Kommunikationspartnern und Anwendung aufgebaut. Jedem Namen der Anwendung ist eine eigene Adresse für den Verbindungsaufbau zugeordnet.

Datenstruktur kc_bcamappl_str
`char bc_name[8];`
`char t_prot[6];`
`char listener_id[5];` (nur auf Unix-, Linux- und Windows-Systemen)
`char listener_port[5];`
`char tsel_format;` (nur auf Unix-, Linux- und Windows-Systemen)
`char signon_tac[8];`
`char secure_soc;`
`char user_auth;`

Die Felder der Datenstruktur haben die folgende Bedeutung:

bc_name

enthält den Namen der lokalen Anwendung, deren Eigenschaften UTM zurückliefert.

t_prot Die Bedeutung der Rückgabe in *t_prot* ist abhängig vom Betriebssystem, auf dem die UTM-Anwendung abläuft.

BS2000-Systeme:

t_prot enthält das Transportprotokoll, das auf den Verbindungen zu Partner-Anwendungen verwendet wird, die über diesen Anwendungsnamen aufgebaut werden.
Es ist nur das erste Feldelement von *t_prot* besetzt. Der Rest ist mit Leerzeichen aufgefüllt.

Das Transportprotokoll wird wie folgt angegeben:

'N' NEA-Transportprotokoll

'I' ISO-Transportprotokoll

'R' ISO Transportprotokoll und RFC1006 Konvergenzprotokoll über TCP/IP-Verbindungen

'TA' TCP/IP-Protokoll mit HTTP- oder USP-Protokoll

'TH' TCP/IP-Protokoll mit HTTP-Protokoll

'TU' TCP/IP-Protokoll mit USP-Protokoll

Unix-, Linux- und Windows-Systeme:

t_prot enthält das Adressformat, das dem BCAMAPPL-Namen bei der KDCDEF-Generierung zugeordnet wurde.

Die Adressformate werden wie folgt angegeben:

'R' ISO Transportprotokoll und RFC1006 Konvergenzprotokoll über TCP/IP-Verbindungen

'TA' TCP/IP-Protokoll mit HTTP- oder USP-Protokoll

'TH' TCP/IP-Protokoll mit HTTP-Protokoll

'TU' TCP/IP-Protokoll mit USP-Protokoll

listener_id (nur auf Unix-, Linux- und Windows-Systemen)

enthält die Listener-Id des BCAMAPPL-Namens. Das ist eine positive ganze Zahl zwischen 0 und 32767.

Mit der Listener-Id wird festgelegt, welche Verbindungen zusammen von demselben Netzprozess verwaltet werden sollen. Alle Verbindungen, die über Zugriffspunkte und BCAMAPPL-Namen mit derselben Listener-Id aufgebaut werden, werden von einem Netzprozess verwaltet.

BCAMAPPL-Namen mit *t_prot*='T' (SOCKET) bilden einen eigenen Nummernkreis. D.h. es werden keine BCAMAPPL-Namen für die Kommunikation über die Socket-Schnittstelle mit BCAMAPPL-Namen/Zugriffspunkten für andere Transportprotokolle in einem Netzprozess zusammengefasst, auch dann nicht, wenn die Listener-Id gleich ist.

listener_port

ist nur relevant bei *t_prot*='T' oder 'R' ('R' nur auf Unix-, Linux- und Windows-Systemen).

listener_port enthält die Portnummer, an der UTM auf Verbindungsanforderungen von außen wartet. Es wird die Portnummer übergeben, die bei der KDCDEF-Generierung angegeben wurde. Siehe dazu auch openUTM-Handbuch „Anwendungen generieren“.

In UTM-Anwendungen auf BS2000-Systemen ist *listener_port* nur belegt, wenn *t_prot*='T' generiert ist. In allen anderen Fällen ist *listener_port*='0'.

In UTM-Anwendungen auf Unix-, Linux und Windows-Systemen bedeutet *listener_port*='0' , dass bei der Generierung keine Portnummer angegeben wurde.

tsel_format (nur auf Unix-, Linux- und Windows-Systemen)

Contains the format indicator of the T-selector in the address.

'T' TRANSDATA-Format

'E' EBCDIC-Zeichenformat

'A' ASCII-Zeichenformat

Enthält *tsel_format* ein Leerzeichen, dann wurde bei der KDCDEF-Generierung kein Formatindikator definiert.

Zur Bedeutung der Adressformate siehe „Dokumentation zu PCMX“ im Abschnitt "openUTM-Dokumentation".

signon_tac

signon_tac enthält entweder den Namen des Transaktionscodes des dieser Anwendung zugeordneten Anmelde-Vorgangs oder ist leer (kein Anmelde-Vorgang).

secure_soc

'N' Die Kommunikation über diese Anwendung erfolgt ohne Verwendung des Secure Socket Layers.

'Y' Die Kommunikation über diese Anwendung erfolgt unter Verwendung des Secure Socket Layers.

user_auth

'N' Als Authentisierungsmechanismus für HTTP-Clients ist *NONE generiert.

'B' Als Authentisierungsmechanismus für HTTP-Clients ist *BASIC generiert.

11.3.1.5 kc_character_set_str - Namen von Character Sets (nur auf BS2000-Systemen)

Für den Objekttyp KC_CHARACTER_SET ist die Datenstruktur *kc_character_set_str* definiert. In *kc_character_set_str* liefert UTM bei KC_GET_OBJECT die Namen und Zuordnungen der für die Anwendung definierten Character Sets zurück.

Datenstruktur kc_character_set_str
`char cs_name[32];`
`char map;`

Die Felder der Datenstruktur haben die folgende Bedeutung:

cs_name

enthält den Namen des Character Sets.

map

enthält die Nummer der Code-Konvertierungstabelle, der der in cs_name angegebene Character Set zugeordnet ist.

11.3.1.6 kc_cluster_node_str - Knoten-Anwendungen einer UTM-Cluster-Anwendung

Für den Objekttyp KC_CLUSTER_NODE ist die Datenstruktur *kc_cluster_node_str* definiert. In *kc_cluster_node_str* liefert openUTM bei KC_GET_OBJECT die Eigenschaften der einzelnen Knoten-Anwendungen (Instanzen) einer UTM-Cluster-Anwendung (Unix-, Linux- und Windows-Systeme) zurück.

mod[1]	Datenstruktur kc_cluster_node_str
-	`char node_indx[4];`
x(GID)	`char hostname[8];`
x(GID)	`struct kc_file_base filebase;`
-	`char bcamappl[8];`
-	`char port_nbr[8];`
-	`struct kc_admi_date_time_model kdcdef_time;`
-	`struct kc_admi_date_time_model startup_time;`
-	`struct kc_admi_date_time_model shut_n_time;`
-	`char start_type;`
-	`char node_state;`
-	`char monitored_node[4];`
-	`char monitoring_node[4];`
-	`struct kc_admi_date_time_model state_change_time;`
x(GID)	`char virtual_host[8];`
-	`node_name[8]`
x(GID)	`char hostname_long[64];`
x(GID)	`char virtual_host_long[64];`

[1] Feldinhalt mit KC_MODIFY_OBJECT modifizierbar; siehe Abschnitt "obj_type=KC_CLUSTER_NODE"

Die Felder der Datenstrukturen haben die folgende Bedeutung:

node_indx

Nummer (Index) der Knoten-Anwendung innerhalb der UTM-Cluster-Anwendung. Die Nummer wird Cluster-intern vergeben und für die Diagnose verwendet. Über den Index ist die Knoten-Anwendung innerhalb der UTM-Cluster-Anwendung eindeutig identifizierbar.
Der Knoten-Index ergibt sich aus der Reihenfolge der CLUSTER-NODE-Statements im KDCDEF-Input: Der Knoten, der durch das erste auftretende Statement beschrieben wird, hat den Index '1', der zweite '2' usw.

KC_MODIFY_OBJECT:
Wenn Sie die Eigenschaften einer Knoten-Anwendung ändern wollen, müssen Sie die Nummer der Knoten-Anwendung im Identifikationsbereich übergeben. Ggf. müssen Sie die Nummer zuvor durch einen KC_GET_OBJECT-Aufruf ermitteln. Sie können nur Knoten modifizieren, die nicht aktiv sind.

hostname

enthält den primären Rechnernamen des Knotens, auf dem diese Knoten-Anwendung abläuft.
Der in diesem Feld zurückgegebene Name ist ggf. auf 8 Zeichen verkürzt. Der vollständige bis zu 64 Zeichen lange Rechnername wird in dem Feld *hostname_long* zurückgegeben.

KC_MODIFY_OBJECT:
Geben Sie den primären Namen des Knotens an, auf dem die Knoten-Anwendung ablaufen soll.

Der Name kann bis zu 8 Zeichen lang sein.

filebase

Basisname der KDCFILE, der Benutzer-Protokolldatei und der System-Protokolldatei SYSLOG der Knoten-Anwendung. Beim Start der Knoten-Anwendung werden die UTM-Systemdateien unter dem hier angegebenen Namen erwartet.

Der Name wird in dem Element *filebase* vom Typ *kc_file_base* übergeben:

struct kc_file_base
`char length[2];`
`char fb_name[42];`

fb_name enthält den Basisnamen, *length* die Länge des Basisnamens.

KC_MODIFY_OBJECT:
Sie können den Basisnamen der Knoten-Anwendung ändern. Dabei ist Folgendes zu beachten:

- Die Basisnamen der einzelnen Knoten-Anwendungen einer UTM-Cluster-Anwendung müssen sich voneinander unterscheiden.
- Geben Sie das Dateiverzeichnis an, das die UTM-Systemdateien der Knoten-Anwendung enthält. Der hier angegebene Name muss aus Sicht aller Knoten das gleiche Dateiverzeichnis bezeichnen. Er kann bis zu 27 Zeichen lang sein.

bcamappl

Name des Transportsystemendpunktes (BCAMAPPL-Name), der für die Clusterinterne Kommunikation verwendet wird. Er wird bei der Generierung in der CLUSTER-Anweisung festgelegt.

port_nbr

Nummer des Listener-Ports, der für die Cluster-interne Kommunikation verwendet wird. Er wird bei der Generierung in der CLUSTER-Anweisung festgelegt.

kdcdef_time

Zeitpunkt, an dem die KDCFILE dieser Knoten-Anwendung generiert wurde.

Datum und Uhrzeit werden in dem Element *kdcdef_time* vom Typ *kc_admi_date_time_model* zurückgeliefert:

struct kc_admi_date_time_model
`struct kc_admi_date_model admi_date;`
`struct kc_admi_time_model admi_time`

wobei

struct kc_admi_date_model
`char admi_day [2];`
`char admi_month [2];`
`char admi_year_4 [4];`
`char admi_julian_day [3];`
`char admi_daylight_saving_time`

und

struct kc_admi_time_model
`admi_hours [2];`
`admi_minutes [2];`
`admi_seconds [2]`

startup_time

Zeitpunkt des letzten Starts dieser Knoten-Anwendung.

Datum und Uhrzeit des Starts werden in dem Element *startup_time* vom Typ *kc_admi_date_time_model* zurückgeliefert (siehe *kdcdef_time*).

shut_n_time

Zeitpunkt, zu dem diese Knoten-Anwendung zuletzt normal beendet wurde.

Datum und Uhrzeit werden in dem Element t *shut_n_time* vom Typ *kc_admi_date_time_model* zurückgeliefert (siehe *kdcdef_time*).

state_change_time;

Zeitpunkt der letzten Status-Änderung dieser Knoten-Anwendung (siehe *node_state*).

Datum und Uhrzeit werden in dem Element *state_change_time* vom Typ *kc_admi_date_time_model* zurückgeliefert (siehe *kdcdef_time*).

start_type

Art des letzten Starts dieser Knoten-Anwendung:

'C' Der letzte Start der Anwendung war ein Kaltstart nach einem normalen Anwendungsende (COLD).

'W' Der letzte Start der Anwendung war ein Warmstart nach einem abnormalen Anwendungsende (WARM).

'D' Die Knoten-Anwendung wurde nach dem Generierungslauf das erste Mal gestartet (DEF).

'U' Die Knoten-Anwendung wurde nach einem KDCUPD-Lauf gestartet (UPDATE).

node_state

Status der Knoten-Anwendung:

'G' (Generated)

Die Knoten-Anwendung wurde nach dem Generierungslauf noch nicht gestartet.

'R' (Running)

Die Knoten-Anwendung läuft zur Zeit.

'T' (Terminated)

Die Knoten-Anwendung läuft nicht. Sie wurde normal beendet.

'A' (Abnormally terminated)

Die Knoten-Anwendung läuft nicht. Sie wurde abnormal beendet.

'F' (Failure)

Die Knoten-Anwendung wurde von ihrer überwachenden Knoten-Anwendung als ausgefallen erkannt.

monitored_node

Nummer (Index) der Knoten-Anwendung, die von dieser Knoten-Anwendung überwacht wird, d.h. deren Verfügbarkeit zyklisch überprüft wird.

monitoring_node

Nummer (Index) der Knoten-Anwendung, die die Verfügbarkeit dieser Knoten-Anwendung überwacht.

virtual_host

Durch die Angabe von HOSTNAME kann die Absenderadresse für Netzverbindungen spezifiziert werden, die von dieser Knoten-Anwendung aus aufgebaut werden.

Leerzeichen bedeuten, dass bei Verbindungsaufbauten die Standard-Absenderadresse des Transportsystems verwendet wird. Diese Funktion wird in einem Rechnerverbund benötigt, wenn beim Verbindungsaufbau als Absenderadresse die relocatable IP-Adresse und nicht die stationäre IP-Adresse verwendet werden soll.

Der in diesem Feld zurückgegebene Name ist ggf. auf 8 Zeichen verkürzt. Der vollständige bis zu 64 Zeichen lange Rechnername wird in dem Feld *virtual_host_long* zurückgegeben.

node_name

Referenzname der Knoten-Anwendung.

Standardwert: NODE*nn*
nn = 01..32, wobei *nn* durch die Reihenfolge der CLUSTER-NODE-Anweisungen bei der Generierung bestimmt wird.

hostname_long

enthält den primären Rechnernamen des Knotens, auf dem diese Knoten-Anwendung abläuft.

KC_MODIFY_OBJECT:
Geben Sie den primären Namen des Knotens an, auf dem die Knoten-Anwendung ablaufen soll.

virtual_host_long

Durch die Angabe von ***virtual_host_long*** kann die Absenderadresse für Netzverbindungen spezifiziert werden, die von dieser Knoten-Anwendung aus aufgebaut werden.

Leerzeichen bedeuten, dass bei Verbindungsaufbauten die Standard- Absenderadresse des Transportsystems verwendet wird. Diese Funktion wird in einem Rechnerverbund benötigt, wenn beim Verbindungsaufbau als Absenderadresse die verschiebbare (relocatable) IP-Adresse und nicht die stationäre IP-Adresse verwendet werden soll.

11.3.1.7 kc_con_str - LU6.1-Verbindungen

Für den Objekttyp KC_CON ist die Datenstruktur *kc_con_str* definiert. In *kc_con_str* liefert UTM bei KC_GET_OBJECT die Eigenschaften und den aktuellen Zustand von Partner-Anwendungen und Verbindungen für die verteilte Verarbeitung über LU6.1 zurück.

Verbindungen für die verteilte Verarbeitung und ihre Eigenschaften können dynamisch erzeugt und gelöscht werden (KC_CREATE_OBJECT Objekttyp KC_CON, KC_DELETE_OBJECT *subopcode1*=KC_IMMEDIATE oder KC_DELAY, Objekttyp KC_CON, siehe auch Abschnitt "KC_DELETE_OBJECT - Objekte löschen").

Datenstruktur kc_con_str
`char co_name[8];`
`char pronam[8];`
`char bcamappl[8];`
`char lpap[8];`
`char termn[2];`
`char listener_port[5];` (nur auf Unix-, Linux- und Windows-Systemen)
`char t_prot;` (nur auf Unix-, Linux- und Windows-Systemen)
`char tsel_format;` (nur auf Unix-, Linux- und Windows-Systemen)
`char state;`
`char auto_connect;`
`char connect_mode;`
`char contime_min[10];`
`char letters[10];`
`char conbad[5];`
`char ip_addr[15];`
`char co_deleted;`
`char ip_addr_v6[39];`
`char ip_v[2];`
`char pronam_long[64];`

Die Felder der Datenstruktur haben die folgende Bedeutung:

co_name

enthält den Namen der Partner-Anwendung, mit der über die logische Verbindung kommuniziert werden soll. Der Name kann maximal 8 Zeichen lang sein.

pronam

enthält den Namen des Rechners, auf dem sich die Partner-Anwendung *co_name* befindet.

In einer UTM-Anwendung auf BS2000-Systemen ist das entweder der Name eines Unix-, Linux- oder Windows-Systems oder eines BS2000-Hosts.

In einer UTM-Anwendung auf Unix-, Linux- oder Windows-Systemen enthält *pronam* den Namen des Partner-Rechners, mit dem UTM die IP-Adresse des Partners im Name-Service sucht.

Wenn der reale Rechnername länger als 8 Zeichen ist, dann gilt Folgendes:

- Das Feld *pronam* enthält einen symbolischen lokalen Namen, der vom Transportsystem für diesen Rechner vergeben wurde.
- Der vollständige, bis zu 64 Zeichen lange Name kann dem Feld *pronam_long* entnommen werden.
- War noch keine Verbindung aufgebaut, dann enthält *pronam* Leerzeichen.

bcamappl

enthält den Namen der lokalen Anwendung, über den die Verbindung zu der Partner-Anwendung aufgebaut wird. *bcamappl* kann der in der KDCDEF-Steueranweisung MAX definierte Anwendungsname (APPLINAME) oder ein BCAMAPPL-Name der Anwendung sein. Der Name ist maximal 8 Zeichen lang.

lpap übergibt den Namen der Partner-Anwendung, zu der die logische Verbindung aufgebaut wird. Dies ist der Name des LPAP-Partners, über den sich die Partner-Anwendung anschließt.

termn enthält das Kennzeichen für die Art des Kommunikationspartners. Das Kennzeichen wird im KB-Kopf für Auftragnehmer-Vorgänge eingetragen, d.h. für solche Vorgänge in der lokalen Anwendung, die von einer Partner-Anwendung gestartet werden. Das Kennzeichen wird vom Anwender definiert und dient dazu die Kommunikationspartner in Gruppen bestimmten Typs einzuteilen. Es wird von UTM nicht ausgewertet.

Das Terminalkennzeichen ist zwei Zeichen lang.

listener_port (nur auf Unix-, Linux- und Windows-Systemen)

enthält die Portnummer der Transportadresse der Partner-Anwendung.

listener_port='0' bedeutet, dass beim Erzeugen des CON-Objekts keine Portnummer angegeben wurde.

t_prot (nur auf Unix-, Linux- und Windows-Systemen)

enthält das Adressformat, mit dem sich die Partner-Anwendung beim Transportsystem anmeldet. Die Adressformate werden wie folgt angegeben:

'R' RFC1006, ISO-Transportprotokoll Klasse 0 über TCP/IP und Konvergenzprotokoll RFC1006.

tsel_format (nur auf Unix-, Linux- und Windows-Systemen)

enthält den Formatindikator des T-Selektors der Partneradresse.

'T' TRANSDATA-Format

'E' EBCDIC-Zeichenformat

'A' ASCII-Zeichenformat

Zur Bedeutung der Adressformate siehe „Dokumentation zu PCMX“ im Abschnitt "openUTM-Dokumentation".

state gibt den Status der Partner-Anwendung bzw. ihres LPAP-Partners an:

'Y' Die Partner-Anwendung ist nicht gesperrt (ON). Die Verbindung zu der Partner-Anwendung besteht oder kann aufgebaut werden.

'N' Die Partner-Anwendung ist gesperrt (OFF). Zu der Partner-Anwendung kann keine logische Verbindung aufgebaut werden.
Damit die Anwendung mit der Partner-Anwendung arbeiten kann, muss die Sperre explizit durch die Administration aufgehoben werden (siehe *kc_lpap_str.state* im Abschnitt "kc_lpap_str - Eigenschaften von LU6.1-Partner-Anwendungen").

auto_connect

gibt an, ob die Verbindung zur Partner-Anwendung beim Start der Anwendung automatisch aufgebaut wird:

'Y' Beim Start der Anwendung versucht UTM, die Verbindung automatisch aufzubauen. D.h. ist die Partner-Anwendung zum Zeitpunkt des Starts der lokalen Anwendung verfügbar, dann steht die Verbindung nach dem Start.

'N' Kein automatischer Verbindungsaufbau beim Start.

connect_mode

gibt den aktuellen Zustand der Verbindung an:

'Y' Die Verbindung ist aufgebaut.

'W' UTM versucht gerade die Verbindung aufzubauen (waiting for connection).

'N' Die Verbindung ist nicht aufgebaut.

contime_min

gibt an, wieviele Minuten die Verbindung zur Partner-Anwendung bereits besteht.

letters

enthält die Anzahl der Ein- und Ausgabe-Nachrichten für die Partner-Anwendungenthält die Anzahl der Ein- und Ausgabe-Nachrichten für die Partner-Anwendung

conbad

gibt an, wie oft die Verbindung seit dem letzten Start der lokalen Anwendung ausgefallen ist.

ip_addr

liefert die von UTM für diese Verbindung verwendete IP-Adresse aus der Objekttabelle der Anwendung, wenn es sich um eine IPv4-Adresse handelt.

BS2000-Systeme:

openUTM liefert im Feld *ip_addr* immer Leerzeichen zurück.

Unix-, Linux- und Windows-Systeme:

Eine IPv6-Adresse wird im Feld *ip_addr_v6* zurückgeliefert (siehe unten).

Die Adresse benutzt UTM, um Verbindungen zur Partner-Anwendung aufzubauen. Die IP-Adresse wird von UTM mit Hilfe des generierten Rechnernamens (*pronam*) beim Anwendungsstart aus dem Name-Service gelesen.
Existiert in den Objekttabellen für den Partner-Rechner keine IPv4-Adresse, dann liefert UTM in *ip_addr* Leerzeichen zurück.

co_deleted

zeigt an, ob die Transportverbindung dynamisch aus der Konfiguration gelöscht wurde:

'Y' die Transportverbindung ist gelöscht.

'N' die Transportverbindung ist nicht gelöscht.

ip_addr_v6

liefert die von UTM für diese Verbindung verwendete IP-Adresse aus der Objekttabelle der Anwendung, wenn es sich um eine IPv6-Adresse handelt oder um eine IPv4-Adresse, die in ein IPv6-Format eingebettet ist.

BS2000-Systeme:

openUTM liefert im Feld *ip_addr_v6* immer Leerzeichen zurück.

Unix- Linux- und Windows-Systeme:

Eine IPv4-Adresse wird im Feld *ip_addr* zurückgeliefert (siehe oben).

Die Adresse benutzt UTM, um Verbindungen zur Partner-Anwendung aufzubauen.
Die IP-Adresse wird von UTM mit Hilfe des generierten Rechnernamens (*pronam*) beim Anwendungsstart aus dem Name-Service gelesen.
Existiert in den Objekttabellen für den Partner-Rechner keine IPv6-Adresse, dann liefert UTM in *ip_addr_v6* Leerzeichen zurück.

ip_v

gibt an, ob es sich bei der von UTM für diese Verbindung verwendeten IP-Adresse um eine IPv4- oder um eine IPv6-Adresse handelt:

Unix-, Linux- und Windows-Systeme:

'V4' IPv4-Adresse.

'V6' IPv6-Adresse oder IPv4-Adresse, die in ein IPv6-Format eingebettet ist.

BS2000-Systeme:

openUTM liefert im Feld *ip_v* immer Leerzeichen zurück.

pronam_long

enthält den Namen des Rechners, auf dem sich die Partner-Anwendung *co_name* befindet.

In einer UTM-Anwendung auf BS2000-Systemen ist das entweder der Name eines Unix-, Linux- oder Windows-Systems oder eines BS2000-Hosts. *pronam_long* is immer belegt.

In einer UTM-Anwendung auf Unix-, Linux- oder Windows-Systemen enthält *pronam_long* den Namen des Partner-Rechners, mit dem UTM die IP-Adresse des Partners im Name Service sucht.

11.3.1.8 kc_db_info_str - Datenbank-Informationen ausgeben

Für den Objekttyp KC_DB_INFO ist die Datenstruktur *kc_db_info_str* definiert. Bei KC_GET_OBJECT liefert UTM in *kc_db_info_str* Informationen über die generierten Datenbank-Anschlüsse zurück.

Mit KC_MODIFY_OBJECT können Sie das Datenbank-Passwort und/oder den Datenbank-Benutzernamen ändern.

Datenbank-Anschlüsse werden mit der KDCDEF-Steueranweisung DATABASE (BS2000-Systeme) oder RMXA (Unix-, Linux- und Windows-Systeme) generiert.

mod[1]	**Datenstruktur kc_db_info_str**
-	`char db_id[2];`
-	`char db_type[8];`
-	`char db_entry_name[8];`
-	`char db_lib_info[54];`
-	`char db_xaswitch[54];`
x(GPD)	`char db_userid[30];`
x(GPD)	`char db_password[30];`
-	`char db_new_userid[30];`

[1] Feldinhalt mit KC_MODIFY_OBJECT modifizierbar; siehe im Abschnitt "obj_type=KC_DB_INFO"

Die Felder der Datenstruktur haben die folgende Bedeutung:

db_id

gibt die Id der Datenbank an. Die Id ist eine Ziffer und repräsentiert die Datenbanken in der Reihenfolge, wie sie generiert wurden. Die Id wird intern durch openUTM vergeben.

db_type

gibt den Typ des Datenbanksytems an:

'XA'
'UDS' (nur auf BS2000-Systemen)
'LEASY' (nur auf BS2000-Systemen)
'SESAM' (nur auf BS2000-Systemen)
'CIS' (nur auf BS2000-Systemen)
'DB' (nur auf BS2000-Systemen)

db_entry_name

- Für eine BS2000-Datenbank wird der Entry-Name der BS2000-Datenbank ausgegeben. Bei einer BS2000-Datenbank mit *db_type*=XA wird in *db_entry_name* der Name des XA-Switches zurückgeliefert, der mit dem ENTRY-Operand der DATABASE-Anweisung generiert wurde. Dieser XA-Switch-Name wird auf einem BS2000-System zusätzlich auch im Feld *db_xaswitch* zurückgegeben.
- Bei Unix-, Linux- und Windows-Systemen enthält *db_entry_name* keine relevante Information.

db_lib_info

Dieses Feld hat eine plattformspezifische Bedeutung.

- Auf BS2000-Systemen entspricht dieses Feld genau dem Parameter LIB der KDCDEF-Anweisung DATABASE, d.h. es wird Information zur Bibliothek ausgegeben, aus welcher der Verbindungsmodul zu dem Datenbanksystem nachgeladen wurde. Hier steht entweder direkt der Name einer Objekt-Modul-Bibliothek oder eine LOGICAL-ID im Sinne der IMON-Installation im Format "LOGICAL-ID (logical-id)".
- Bei Unix-, Linux- und Windows-Systemen enthält dieses Feld den internen Namen des geladenen XA-Switches, z.B. "Oracle_XA".

db_xaswitch

Dieses Feld hat eine plattformspezifische Bedeutung:

- Auf BS2000-Systemen wird in *db_xaswitch* der Inhalt von *db_entry_name* zurückgegeben.
- Bei Unix-, Linux- und Windows-Systemen enthält dieses Feld den Namen des XA-Switches des Resource-Managers. Dieser Name wird im Parameter XASWITCH der KDCDEF-Anweisung RMXA festgelegt.

db_userid

Für XA-Datenbanken gibt UTM bei KC_GET_OBJECT im Feld *db_userid* den für dieses Datenbanksystem generierten Benutzernamen zurück.

Für eine XA-Datenbank kann der Datenbank-Benutzername in diesem Feld per KC_MODIFY_OBJECT auch geändert werden. Die Änderung wirkt immer erst beim nächsten Start der Anwendung.

db_password

Im Feld *db_password* kann für eine XA-Datenbank das Datenbank-Passwort mit KC_MODIFY_OBJECT neu zugewiesen werden.

Bei KC_GET_OBJECT wird dieses Feld von UTM immer mit Leerzeichen versorgt.

db_new_userid

Für XA-Datenbanken gibt UTM bei KC_GET_OBJECT im Feld *db_new_userid* den für dieses Datenbanksystem modifizierten aber noch nicht aktivierten Benutzernamen zurück. Der hier zurückgegebene Benutzername wird beim nächsten Start der Anwendung aktiviert und an das Datenbanksystem übergeben.

11.3.1.9 kc_edit_str - Optionen von EDIT-Profilen (BS2000-Systeme)

Für den Objekttyp KC_EDIT ist die Datenstruktur *kc_edit_str* definiert. Bei KC_GET_OBJECT liefert UTM in *kc_edit_str* Informationen über EDIT-Profile zurück.

EDIT-Profile werden mit der KDCDEF-Steueranweisung EDIT generiert. In EDIT-Profilen werden Bildschirmfunktionen und Eigenschaften der Bildschirmausgabe im Zeilenmodus zusammengefasst. Jedem EDIT-Profil wird bei der KDCDEF-Generierung ein Name zugeordnet, über den der zugehörige Satz von Editoptionen aus einem Teilprogrammlauf heraus angesprochen werden kann.

Eine ausführlichere Beschreibung der im Folgenden angegebenen Editoptionen finden Sie im Benutzerhandbuch TRANSDATA TIAM. Nähere Informationen zum Arbeiten mit EDIT-Profilen finden Sie im openUTM-Handbuch „Anwendungen programmieren mit KDCS“.

Datenstruktur kc_edit_str
`char ed_name[8];`
`char edit_mode;`
`char edit_bell;`
`char hcopy;`
`char hom;`
`char ihdr;`
`char locin;`
`char low;`
`char nolog;`
`char ohdr;`
`char saml;`
`char specin;`
`char ccsname[8];`

Die Felder der Datenstruktur haben die folgende Bedeutung:

ed_name

enthält den Namen des EDIT-Profils, dessen Editoptionen UTM zurückliefert. Es ist ein bis zu 7 Zeichen langer alphanumerischer Name.

edit_mode

gibt an, in welchem Modus die Nachrichten ausgegeben werden:

'E' (extended linemode)
Die Nachrichten werden im „extended linemode“ ausgegeben.

'I' (info)
Die Nachrichten können in einer speziellen Informationszeile (Systemzeile) abgebildet werden, ohne dass dabei wichtige Daten am Terminal überschrieben werden.

'L' (line mode)
Die Nachricht wird im Zeilenmodus ausgegeben. Sie kann durch logische Steuerzeichen strukturiert werden. Die Nachricht wird vom System aufbereitet.

'P' (physical mode)
Die Nachricht wird physikalisch, d.h. ohne Aufbereitung durch das System ausgegeben bzw. eingelesen.

'T' (transparent mode)
Die Ausgabe-Nachricht wird transparent übertragen.

edit_bell

gibt an, ob bei der Ausgabe der Nachricht am Terminal ein akustischer Alarm ausgelöst wird. Der Feldinhalt bedeutet:

'Y' Es wird akustischer Alarm ausgelöst.

'N' Es wird kein akustischer Alarm ausgelöst.

hcopy (**h**ard **copy**)
gibt an, ob eine Ausgabe-Nachricht zusätzlich zur Ausgabe am Terminal auch auf einem dort angeschlossenen Hardcopy-Drucker protokolliert wird.

'Y' Protokollierung der Ausgabe-Nachricht auf Hardcopy-Drucker

'N' keine Protokollierung

hom (**hom**ogeneous)
gibt an, ob die Ausgabe-Nachricht unstrukturiert, d.h. homogen, ausgegeben wird.

'Y' Die Nachricht wird unstrukturiert ausgegeben.

'N' Die Nachricht wird strukturiert ausgegeben. In diesem Fall wird eine logische Zeile als Ausgabeeinheit betrachtet.

ihdr (**i**nput **h**ea**d**e**r**)
gibt an, ob der Nachrichtenvorspann der Eingabe-Nachricht an das Teilprogramm übergeben wird.

'Y' Der Nachrichtenvorspann der Eingabe-Nachricht wird übergeben

'N' Der Nachrichtenvorspann wird nicht übergeben.

locin (**loc**al **in**put parameter)
gibt an, ob lokale Attribute in der Eingabe-Nachricht als logische Steuerzeichen an den Anwender weitergereicht werden.

'Y' lokale Attribute in der Eingabe-Nachricht werden als logische Steuerzeichen weitergereicht.

'N' lokale Attribute werden entfernt und nicht weitergereicht.

low (**low**er case)
gibt an, ob die Eingabe-Nachricht, die an das Teilprogramm übergeben wird, auch Kleinbuchstaben enthalten darf.

'Y' Kleinbuchstaben in der Eingabe-Nachricht werden an das Teilprogramm übergeben.

'N' Kleinbuchstaben werden vor der Übergabe an das Teilprogramm in Großbuchstaben umgesetzt.

nolog (**no log**ical characters)
gibt an, wie nicht abdruckbare Zeichen von dem System behandelt werden.

'Y' Die logischen Steuerzeichen werden nicht ausgewertet. Alle Zeichen, die im EBCDIC-Code kleiner X'40' sind, werden durch Ersatzzeichen (SUB) ersetzt. Nur abdruckbare Zeichen werden durchgelassen.

'N' Alle logischen Steuerzeichen werden ausgewertet. Spezielle physikalische Steuerzeichen werden durchgelassen. Andere Zeichen kleiner X'40' werden durch Ersatzzeichen (SUB) ersetzt. Abdruckbare Zeichen werden durchgelassen.

ohdr (**o**utput **h**ea**d**e**r**)
gibt an, ob die Ausgabe-Nachricht einen Nachrichtenkopf enthält. Die Länge des Nachrichtenkopfs +1 wird im ersten Byte der Nachricht binär angegeben.

'Y' Die Ausgabe-Nachricht enthält Nachrichtenkopf.

'N' Die Ausgabe-Nachricht enthält keinen Nachrichtenkopf.

saml (**sam**e **l**ine)
gibt an, ob am Nachrichtenanfang ein Zeilenvorschub unterdrückt wird. Der Inhalt von saml ist nur für Drucker von Bedeutung. Der Feldinhalt von saml hat folgende Bedeutung:

'Y' Am Nachrichtenanfang wird kein Zeilenvorschub ausgeführt.

'N' Die Nachricht beginnt am Anfang der nächsten Zeile.

specin (**spec**ial **in**put)

gibt an, welche speziellen Optionen das Editprofil für die Eingabe enthält.

'C' (confidential)
Die Eingabedaten werden am Terminal dunkel dargestellt.

'I' (ID card)
Die nächste Eingabe erfolgt über den Ausweisleser.

'N' (normal)
Es erfolgt eine normale Eingabe vom Terminal.

ccsname (**c**oded **c**haracter **s**et **name**)

enthält den Namen des Zeichensatzes (CCS-Name), der für die Aufbereitung einer Nachricht verwendet wird (siehe auch Benutzerhandbuch zu XHCS).

11.3.1.10 kc_gssb_str - GSSBs der Anwendung

Für den Objekttyp KC_GSSB ist die Datenstruktur *kc_gssb_str* definiert. Bei KC_GET_OBJECT liefert UTM in *kc_gssb_str* die Namen von aktuell in der Anwendung existierenden Globalen Sekundär-Speicherbereichen (GSSB) zurück. Ein GSSB wird von KDCS-Teilprogrammen zur Übergabe von Daten zwischen Vorgängen verwendet.

Datenstruktur kc_gssb_str
`char gs_name[8];`

Das Feld hat die folgende Bedeutung:

gs_name

enthält den Namen des GSSBs.

11.3.1.11 kc_http_descriptor_str - HTTP-Deskriptoren der Anwendung

Für den Objekttyp KC_HTTP_DESCRIPTOR ist die Datenstruktur *kc_http_descriptor_str* definiert. Bei KC_GET_OBJECT liefert UTM in *kc_http_descriptor_str* die Namen und Eigenschaften der für die Anwendung definierten HTTP-Deskriptoren zurück.

Datenstruktur kc_http_descriptor_str
`char hd_name[8];`
`char bcamappl[8];`
`char tac[8];`
`char user_auth;`
`char convert_text;`
`char http_exit[32];`
`char path[254];`

Die Felder der Datenstruktur haben die folgende Bedeutung:

hd_name

enthält den Namen des HTTP-Deskriptors. Dieser Name hat nur anwendungs-lokale Bedeutung.

bcamappl

enthält den Namen der Anwendung, für die dieser HTTP-Deskriptor gilt. Enthält bcamappl den Wert *ALL, dann gilt dieser HTTP-Deskriptor für alle HTTP-Verbindungen der UTM-Anwendung.

tac

enthält den Namen des TACs, der für HTTP-Requests mit dem für diesen HTTP-Deskriptor spezifizierten Path aufgerufen wird.

user_auth

in diesem Feld wird angezeigt, welchen Authorisierungsmechanismus Clients für HTTP-Requests verwenden müssen, die diesem HTTP-Deskriptor zugeordnet werden.

'B' Es wird der Basic Authorisierungsmechanismus verwendet, bei dem der Client UserId und Passwort Base64-codiert in dem Authorization Header des Request mitschicken muss.

'N' Es wird keine Authentisierungsinformation verlangt. Von UTM wird der Verbindungs-User als User verwendet, es sei denn der Client liefert von sich aus Authentisierungsinformation.

convert_text

in diesem Feld wird ausgegeben, ob UTM für Textnachrichten eine Code-Konvertierung durchführen soll oder nicht. convert_text ist nur in BS2000-Systemen von Bedeutung, da auf offenen Systemen keine Code-Konvertierung erforderlich ist.
Eine Code-Konvertierung für den Message Body einer HTTP-Nachricht führt UTM nur durch, wenn

1. der Content-Type Header einer HTTP-Nachricht eine Nachricht vom Typ "text" anzeigt, und
2. sich der im Content-Type Header angezeigte Character Set einem für die UTM-Anwendung definierten CHAR-SET zuordnen lässt.

Die Code-Tabelle, die mit einem auf diese Weise zugeordneten CHAR-SET definiert ist, wird dabei von UTM zur Code-Konvertierung herangezogen.

'Y' UTM soll für Textnachrichten, die diesem HTTP-Deskriptor zugeordnet werden können, eine Code-Konvertierung durchführen.

'N' UTM soll für Textnachrichten, die diesem HTTP-Deskriptor zugeordnet werden können, keine Code-Konvertierung durchführen.

Für UTM auf Unix-, Linux- und Windows-Systemen wird immer "N" ausgegeben.

http_exit

in diesem Feld wird der Name des HTTP-Exit Programms ausgegeben, das von UTM zur Umformatierung der Ein- und Ausgabenachrichten aufgerufen werden soll.

Wenn diesem HTTP-Deskriptor kein HTTP-Exit Programm zugeordnet ist, dann enthält das Feld entsprechend der Angabe bei der Generierung entweder "*NONE" oder "*SYSTEM".

path

in diesem Feld wird der Path ausgegeben, der für diesen HTTP-Deskriptor definiert ist. HTTP-Requests, die diesen Path enthalten bzw. deren Path mit der hier ausgegebenen Zeichenfolge beginnt, werden anhand der Angaben in diesem HTTP-Deskriptor weiter verarbeitet.

11.3.1.12 kc_kset_str - Keysets der Anwendung

Für den Objekttyp KC_KSET ist die Datenstruktur *kc_kset_str* definiert. Bei KC_GET_OBJECT liefert UTM in *kc_kset_str* Informationen zu einem Keyset zurück.
In einem Keyset sind Key- oder Zugangscodes der Anwendung, die für den Zugriffsschutz definiert wurden, zu einem logischen Keyset zusammengefasst.

Ein Keyset kann einem Benutzer, einem LTERM-Partner, einem LTERM-Pool, einem (OSI-)LPAP-Partner oder einer Access Liste zuordnet sein, die den Zugriff z.B. auf TAC-Objekte regelt. Die mit den Keysets verbundenen Zugriffsrechte stehen den Clients oder der Partner-Anwendung nach dem Aufbau der logischen Verbindung bzw. dem Benutzer nach dem Anmelden bei der Anwendung zur Verfügung (siehe auch openUTM-Handbuch „Konzepte und Funktionen“).

Keysets können dynamisch mit KC_CREATE_OBJECT erzeugt, mit KC_DELETE_OBJECT gelöscht oder mit KC_MODIFY_OBJECT modifiziert werden. Welches Keyset einem Client, einer Partner-Anwendung oder einem Benutzer zugeordnet ist, wird in der jeweiligen Datenstruktur des Objekts im Feld *kset* zurückgeliefert.

KDCDEF erzeugt implizit das Keyset KDCAPLKS, das bereits sämtliche Keycodes enthält.

mod[1]	**Datenstruktur kc_kset_str**
-	`char ks_name[8];`
-	`char master;`
x(GPD)	`char keys[4000];`
-	`char ks_deleted;`

[1] Feldinhalt mit KC_MODIFY_OBJECT modifizierbar; siehe im Abschnitt "obj_type=KC_KSET"

Die Felder der Datenstruktur haben die folgende Bedeutung:

ks_name

enthält den Namen des Keysets. Er wird beim Erzeugen des Keysets mit KC_CREATE_OBJECT Objekttyp KC_KSET oder bei der KDCDEF-Generierung in KSET festgelegt.

master

gibt an, ob es sich um ein Master-Keyset handelt. Ein Master-Keyset enthält alle Key- oder Zugangscodes, die für den Zugriff auf die Objekte der Anwendung notwendig sind, d.h. alle Keycodes zwischen 1 und dem Maximalwert, der bei der KDCDEF-Generierung in MAX KEYVALUE festgelegt wurde.

'Y' Das Keyset ist ein Master-Keyset.

'N' Das Keyset ist kein Master-Keyset (Standard).

keys gibt die Key- oder Zugangscodes an, die zu dem Keyset gehören.

Ein Key- oder Zugangscode ist eine ganze Zahl zwischen 1 und dem Wert KEYVALUE, der bei der KDCDEF-Generierung in der MAX-Anweisung festgelegt wurde. KEYVALUE ist der größtmögliche Key- bzw. Zugangscode der Anwendung.
KEYVALUE kann zwischen 1 und 4000 liegen.

keys besteht aus 4000 Feldelementen *keys*[0] bis *keys*[3999]. Der Inhalt der Feldelemente ist folgendermaßen zu interpretieren:

keys[0] =

'0': Der Key- / Zugangscode 1 gehört nicht zu diesem Keyset.
'1': Der Key- / Zugangscode 1 gehört zu diesem Keyset.

keys[n] =

'0': Der Key- / Zugangscode n+1 gehört nicht zu diesem Keyset.
'1': Der Key- / Zugangscode n+1 gehört zu diesem Keyset.

keys[3999] =

'0': Der Key- / Zugangscode 4000 gehört nicht zu diesem Keyset.
'1': Der Key- / Zugangscode 4000 gehört zu diesem Keyset.

ks_deleted

zeigt an, ob das Keyset dynamisch aus der Konfiguration gelöscht wurde:

'Y' das Keyset ist gelöscht.

'N' das Keyset ist nicht gelöscht.

11.3.1.13 kc_load_module_str - Lademodule (BS2000-Systeme) bzw. Shared Objects/DLLs (Unix-, Linux- und Windows-Systeme)

Für den Objekttyp KC_LOAD_MODULE ist die Datenstruktur *kc_load_module_str* definiert. In *kc_load_module_str* liefert UTM bei KC_GET_OBJECT Folgendes zurück:

- BS2000-Systeme:Informationen über die Lademodule, die mit der KDCDEF-Steueranweisung LOAD-MODULE generiert wurden.
- Unix-, Linux- und Windows-Systeme: Informationen über die Shared Objects oder DLLs, die mit der KDCDEF-Steueranweisung SHARED-OBJECT generiert wurden.

Mit einem KDCADMI-Aufruf mit Operationscode KC_MODIFY_OBJECT und Objekttyp KC_LOAD_MODULE können Sie einzelne Lademodule, bzw. Shared Objects oder DLLs während des Anwendungslaufs austauschen.

mod[1]	**Datenstruktur kc_load_module_str**
-	`char lm_name[32];`
x(GID)	`char version[24];`
-	`char lib[54];`
-	`char load_mode;`
-	`char poolname[50];` (nur auf BS2000-Systemen)
-	`char version_prev[24];`
-	`char changeable;`
-	`char change_necessary;` (nur auf BS2000-Systemen)
-	`char altlib;` (nur auf BS2000-Systemen)
-	`char version_gen[24];`

[1] Feldinhalt mit KC_MODIFY_OBJECT modifizierbar; siehe im Abschnitt "obj_type=KC_LOAD_MODULE"

lm_name

enthält den Namen des Lademoduls, Shared Objects bzw. DLLs.

version

in *version* liefert UTM die Versionsnummer des Lademoduls, Shared Objects bzw. DLLs zurück, die zur Zeit geladen ist oder geladen wird. Konnte das Lademodul in der Bibliothek nicht gefunden werden, dann ist *version* mit Leerzeichen belegt.

Nur auf BS2000-Systemen:

*HIGHEST-EXISTING

Bei KC_MODIFY_OBJECT wird die höchste in der Bibliothek vorhandene Version geladen.

*UPPER-LIMIT (bzw. @)

Bei KC_MODIFY_OBJECT wird das Lademodul geladen, das als letztes ohne explizite Versionsangabe in die PLAM-Bibliothek gebracht wurde.

lib der Inhalt von *lib* hat folgende Bedeutung:

- In UTM-Anwendungen auf BS2000-Systemen liefert UTM in *lib* die Programmbibliothek zurück, aus der das Lademodul nachgeladen wird.
- In UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen enthält *lib* das Dateiverzeichnis, in dem das Shared Object/DLL abgelegt ist.

load_mode

enthält den Lademodus des Lademoduls, Shared Objects bzw. DLLs. Der Lademodus gibt an, wann und wohin ein Lademodul/Shared Object/DLL geladen wird.

'U' (STARTUP)
Das Lademodul/Shared Object/DLL wird beim Start der Anwendung als eigenständige Einheit nachgeladen.

Nur auf BS2000-Systemen:

- Beim Nachladen eines Lademoduls werden Externverweise aus allen bereits geladenen Modulen der UTM-Anwendung, aus allen nicht privilegierten Subsystemen und aus dem Klasse-4-Speicher befriedigt.

'O' (ONCALL)
Das Lademodul/Shared Object/DLL wird als eigenständige Einheit nachgeladen, wenn eines seiner Teilprogramme oder seiner VORGANG-Exits erstmalig aufgerufen wird.

Nur auf BS2000-Systemen:

- Beim Nachladen eines Lademoduls werden Externverweise aus allen bereits geladenen Modulen der UTM-Anwendung, aus allen nicht privilegierten Subsystemen und aus dem Klasse-4-Speicher befriedigt.
- Wird mit mehreren Prozessen gearbeitet, so darf während des Anwendungslaufs dieses Lademodul in der Bibliothek (LIB=...) nicht überschrieben werden. Es könnten sonst eventuell unterschiedliche Stände des Lademoduls in einem Anwendungslauf zum Ablauf kommen.

'S' (STATIC, nur auf BS2000-Systemen)
Das Lademodul ist statisch in das Anwendungsprogramm eingebunden.
Das Lademodul kann nicht während eines Anwendungslaufs ausgetauscht werden.

'P' (POOL, nur auf BS2000-Systemen)
Das Lademodul wird beim Start der Anwendung in einen Common Memory Pool (siehe *poolname*) geladen. Das Lademodul besteht nur aus einer public Slice (keine private Slice).

'T' (POOL/STARTUP, nur auf BS2000-Systemen)
Die public Slice des Lademoduls wird beim Start der Anwendung in einen Common Memory Pool (siehe *poolname*) geladen. Die zu dem Lademodul gehörige private Slice wird anschließend in den prozesslokalen Speicher geladen (private Slice mit Lademodus STARTUP).

'C' (POOL/ONCALL, nur auf BS2000-Systemen)
Die public Slice des Lademoduls wird beim Start der Anwendung in einen Common Memory Pool (siehe *poolname*) geladen. Die zu dem Lademodul gehörige private Slice wird in den prozesslokalen Speicher geladen, wenn das erste Teilprogramm aufgerufen wird, das diesem Lademodul zugeordnet ist (private Slice mit Lademodus ONCALL).

poolname (nur auf BS2000-Systemen

wird nur versorgt, wenn das Lademodul bzw. seine public Slice in einen Common Memory Pool geladen werden (*load_mode*='P', 'T' oder 'C'). *poolname* enthält dann den Namen des Common Memory Pools. Der Name kann bis zu 50 Zeichen lang sein.

version_prev

enthält die Vorgängerversion des Lademoduls/Shared Objects/DLLs, d.h. die Version, die vor dem letzten Programmaustausch geladen war.

Wurde das Lademodul/Shared Object/DLL noch nicht ausgetauscht bzw. ist es nicht austauschbar, dann ist *version_prev* mit Leerzeichen belegt.

changeable

gibt an, ob das Lademodul/Shared Object/DLL ausgetauscht werden kann.

'Y' Das Lademodul/Shared Object/DLL kann während des Anwendungslaufs ausgetauscht werden.

'N' Das Lademodul/Shared Object/DLL kann nicht während des Anwendungslaufs ausgetauscht werden.

change_ necessary (nur auf BS2000-Systemen)

ist nur relevant bei Lademodulen, die entweder ganz in einem Common Memory Pool liegen oder deren public Slice im Common Memory Pool liegt. *change_necessary* gibt an, ob dieses Lademodul für einen Programmaustausch vorgemerkt wurde.

Lademodule im Common Memory Pool müssen zunächst mit KC_MODIFY_OBJECT für den Programmaustausch vorgemerkt werden. Danach muss mit KC_CHANGE_APPLICATION der eigentliche Austausch durchgeführt werden.

'Y' Das Lademodul ist für einen Austausch vorgemerkt. Ein Programmaustausch mit KC_CHANGE_APPLICATION ist nötig, um das Lademodul auszutauschen.

'N' Das Lademodul ist nicht für einen Austausch vorgemerkt.

altlib (nur auf BS2000-Systemen)

gibt an, ob das Lademodul mit der Autolink-Funktion des BLS geladen wird.

'Y' Laden mit Autolink

'N' Laden ohne Autolink

version_gen

enthält die Version, mit der das Lademodul/Shared Object/DLL generiert wurde.

11.3.1.14 kc_lpap_str - Eigenschaften von LU6.1-Partner-Anwendungen

Für den Objekttyp KC_LPAP ist die Datenstruktur *kc_lpap_str* definiert. In *kc_lpap_str* liefert UTM bei KC_GET_OBJECT die Eigenschaften eines LPAP-Partners zurück.

Ein LPAP-Partner ist ein logischer Anschlusspunkt für eine LU6.1-Partner-Anwendung. LPAP-Partner werden bei der statischen Generierung mit KDCDEF erzeugt. Die Zuordnung zu einer realen Partner-Anwendungen können Sie bei der Generierung oder dynamisch beim Erzeugen eines neuen CON-Objekts vornehmen.

mod[1]	Datenstruktur kc_lpap_str
-	`char lp_name[8];`
-	`char kset[8];`
-	`char lnetname[8];`
-	`char netprio;` (nur auf BS2000-Systemen)
-	`char permit;`
-	`char qlev[5];`
-	`char rnetname[8];`
x(GPD)	`char state;`
x(GPD)	`char auto_connect;`
-	`char contwin;`
-	`char dpn[8];`
x(GPD)	`char idletime_sec[5];`
-	`char map;` (nur auf Unix-, Linux- und Windows-Systemen)
-	`char paccnt[2];`
-	`char plu;`
x(A)	`char connect_mode;`
x(IR)	`char quiet_connect;`
x(IR)	`char bcam_trace;`
-	`char out_queue[5];`
-	`char nbr_dputs[10];`
-	`char master[8];`
-	`char bundle;`
-	`char out_queue_ex[10];`
x(GPD)	`char dead_letter_q;`

[1] Feldinhalt mit KC_MODIFY_OBJECT modifizierbar; siehe im Abschnitt "obj_type=KC_LPAP"

Die Felder der Datenstruktur haben die folgende Bedeutung:

lp_name
enthält den Namen des LPAP-Partners, d.h. den logischen Namen der Partner-Anwendung. Über diesen Namen spricht die lokale Anwendung die Partner-Anwendung an. *lp_name* hat nur in der lokalen Anwendung eine Bedeutung.

kset
enthält den Namen des Keysets, das der Partner-Anwendung zugeordnet wird. Das Keyset legt die Zugriffsrechte der Partner-Anwendung innerhalb der lokalen Anwendung fest. Die Partner-Anwendung darf nur die Transaktionscodes verwenden, die entweder nicht durch einen Lockcode gesichert sind oder die durch einen Lockcode gesichert sind, zu dem das Keyset den passenden Key- oder Zugangscode enthält.

lnetname
ist nur bei heterogener Kopplung relevant.

lnetname enthält den Namen der lokalen UTM-Anwendung, unter dem diese in der Partner-Anwendung bekannt ist.

netprio
(nur auf BS2000-Systemen)
enthält die Transportpriorität, die auf der diesem LPAP-Partner zugeordneten Transportverbindung benutzt wird.

'M' Transportpriorität „Medium“

'L' Transportpriorität „Low“

permit
gibt an, welche Berechtigungen die Partner-Anwendung innerhalb der lokalen

'A' (ADMIN)
Die Partner-Anwendung hat Administrationsberechtigung, sie darf alle Administrationsfunktionen in der lokalen Anwendung ausführen.

'N' (NONE)
Die Partner-Anwendung hat keine Administrationsberechtigung.

Nur auf BS2000-Systemen:

- Ist die lokale Anwendung eine UTM-Anwendung auf einem BS2000-System, dann darf die Partner-Anwendung auch keine UTM-SAT-Administrationsfunktionen ausführen.

'B' (BOTH, nur auf BS2000-Systemen)
Die Partner-Anwendung darf sowohl Administrations- als auch UTM-SAT-Administrationsfunktionen in der lokalen Anwendung ausführen.

'S' (SAT nur auf BS2000-Systemen)
Die Partner-Anwendung hat UTM-SAT-Administrationsberechtigung. Sie darf Preselection-Funktionen in der lokalen Anwendung ausführen, d.h. sie kann die SAT-Protokollierung bestimmter Ereignisse ein- bzw. ausschalten.

qlev
(**q**ueue **lev**el)
qlev gibt an, wieviele Asynchron-Nachrichten maximal in der lokalen Message Queue für die Partner-Anwendung stehen dürfen. Wird dieser Schwellwert überschritten, so werden weitere Asynchron-Aufträge an die Partner-Anwendung abgewiesen (d.h. bei folgenden APRO-AM-Aufrufen wird '40Z' zurückgeliefert).

rnetname — ist nur bei heterogener Kopplung relevant.

rnetname enthält den VTAM-Namen der Partner-CICS- bzw. IMS-Anwendung.

state — Contains the status of the LPAP partner:

'Y' Der LPAP-Partner ist nicht gesperrt. Es kann eine Verbindung zur Partner-Anwendung aufgebaut werden oder es besteht bereits eine Verbindung.

'N' Der LPAP-Partner ist gesperrt. Es kann keine Verbindung zur Partner-Anwendung aufgebaut werden.

auto_connect — gibt an, ob die lokale Anwendung beim Anwendungsstart automatisch die Verbindung zur Partner-Anwendung aufbaut.

'N' Die Verbindung wird nicht automatisch aufgebaut, sie muss vom Administrator aufgebaut werden.

'Y' Beim Start der lokalen Anwendung wird die Verbindung zur Partner-Anwendung von UTM automatisch aufgebaut, sofern die Partner-Anwendung zu diesem Zeitpunkt verfügbar ist.

Ist in beiden Anwendungen (lokale Anwendung und Partner-Anwendung) der automatische Verbindungsaufbau definiert, dann wird die Verbindung zwischen beiden automatisch aufgebaut, sobald beide Anwendungen verfügbar sind.

contwin — (**cont**ention **win**ner)
gibt an, ob die Partner-Anwendung Contention Winner auf der Session zwischen lokaler Anwendung und Partner-Anwendung ist. Der Contention Winner verwaltet die Session und regelt die Belegung der Session durch Aufträge.

'Y' Die Partner-Anwendung ist Contention Winner.

'N' Die lokale Anwendung ist Contention Winner.

In jedem Fall können jedoch Aufträge sowohl von der lokalen als auch von der Partner-Anwendung gestartet werden. Im Konfliktfall, wenn gleichzeitig die lokale und die Partner-Anwendung einen Auftrag starten wollen, wird die Session mit dem Auftrag des Contention Winners belegt.

dpn — (**d**estination **p**rocess **n**ame)
ist nur bei Kopplung zu IBM-Systemen von Bedeutung.

dpn enthält den Namen der Instanz, die Asynchron-Nachrichten bearbeitet.

idletime_sec — enthält die Zeit in Sekunden, die sich eine Session zur Partner-Anwendung maximal im Leerlauf-Zustand (Idle-Zustand) befinden darf, bevor UTM die Verbindung zur Partner-Anwendung abbaut. Leerlauf-Zustand heißt: die Session ist nicht durch einen Auftrag belegt.

idletime_sec='0' bedeutet, dass der Leerlauf-Zustand nicht überwacht wird.

Minimalwert: '60'
Maximalwert: '32767'

map

(nur auf Unix-, Linux- und Windows-Systemen)
gibt an, ob UTM für Benutzer-Nachrichten ohne Format-Kennzeichen, die zwischen den Partner-Anwendungen ausgetauscht werden, eine Code-Konvertierung (ASCII <-> EBCDIC) durchführt.

'U' (USER)
UTM konvertiert die Benutzer-Nachrichten nicht, d.h. die Daten der Nachricht werden unverändert an die Partner-Anwendung übertragen.

'1', '2', '3', '4' (SYS1 | SYS2 | SYS3 | SYS4)

UTM konvertiert die Benutzernachrichten gemäß den für die Code-Konvertierung bereitgestellten Konvertierungstabellen, d.h.:

- Vor dem Senden wird von ASCII nach EBCDIC konvertiert.
- Nach dem Empfangen wird von EBCDIC nach ASCII konvertiert.

Dabei geht UTM davon aus, dass die Nachrichten nur abdruckbare Zeichen enthalten.

Weitere Informationen zur Code-Konvertierung finden Sie im openUTM-Handbuch „Anwendungen programmieren mit KDCS“, Abschnitt "Code-Konvertierung".

paccnt (**pac**ing **Count**)
enthält die Anzahl der Teilnachrichten, die die lokale Anwendung bei längeren Nachrichten maximal unquittiert empfangen kann.

Ein zu großer Pacing-Wert in *paccnt* kann zu Stauproblemen im Netz führen.
Bei *paccnt*='0' ist die Anzahl der unquittierten Teilnachrichten unbeschränkt.

plu (**p**rimary **l**ogical **u**nit)
gibt an, ob die Partner-Anwendung für den Aufbau der Session zuständig ist, d.h. ob die Partner-Anwendung die 'primary logical unit' (PLU) ist.

'Y' Die Partner-Anwendung ist die 'primary logical unit'.

'N' Die lokale Anwendung ist die 'primary logical unit'.

connect_mode gibt den Status der Verbindung zur Partner-Anwendung an.

'Y' Die Partner-Anwendung ist z.Zt. mit der Anwendung verbunden.

'N' Die Partner-Anwendung ist z.Zt. nicht mit der Anwendung verbunden.

'W' UTM versucht gerade eine Verbindung zur Partner-Anwendung aufzubauen (WAIT).

quiet_connect gibt an, ob für die Verbindung zu dem LPAP-Partner die Eigenschaft QUIET gesetzt ist. QUIET bedeutet, dass UTM die Verbindung zur Partner-Anwendung abbaut, sobald die für die Partner-Anwendung generierten Sessions nicht mehr durch Aufträge belegt sind. Für die Partner-Anwendung werden keine neuen Dialog-Aufträge mehr angenommen.

'Y' Die Eigenschaft QUIET ist gesetzt.

'N' Die Eigenschaft QUIET ist nicht gesetzt.

bcam_trace

gibt an, ob der BCAM-Trace explizit für den LPAP-Partner der Partner-Anwendung eingeschaltet ist oder nicht. Als BCAM-Trace wird die Tracefunktion bezeichnet, die Verbindungs-spezifische Aktivitäten innerhalb einer UTM-Anwendung verfolgt (z.B. BCAM-Tracefunktion auf BS2000-Systemen). Der BCAM-Trace kann allgemein für alle Verbindungen der Anwendung (d.h. für alle LPAP- und LTERM-Partner) eingeschaltet werden oder explizit für bestimmte LTERM- oder LPAP-Partner.

'Y' Der BCAM-Trace ist explizit für diesen LPAP-Partner eingeschaltet worden.
Ist der BCAM-Trace allgemein für alle Verbindungen der UTM-Anwendung eingeschaltet, dann wird in *bcam_trace* 'N' zurückgeliefert.
Ob der BCAM-Trace allgemein eingeschaltet ist, können Sie z.B. durch einen KC_GET_OBJECT-Aufruf mit Parametertyp KC_DIAG_AND_ACCOUNT_PAR ermitteln. Es wird dann *bcam_trace*='Y' in *kc_diag_and_account_par_str* zurückgeliefert.

'N' Der BCAM-Trace ist nicht explizit für diesen LPAP-Partner eingeschaltet worden.

Sie können den BCAM-Trace während des Anwendungslaufs ein- und ausschalten.

out_queue Anzahl der Nachrichten, die zur Zeit in der lokalen Message Queue der Partner-Anwendung zwischengespeichert sind und noch an die Partner-Anwendung gesendet werden müssen.

Ist die Anzahl der Nachrichten größer als 99999, wird die Zahl nur unvollständig dargestellt. Deshalb sollte das Feld *out_queue_ex* verwendet werden, weil hier auch größere Zahlen vollständig abgebildet werden.

nbr_dputs Anzahl der anstehenden zeitgesteuerten Aufträge für dieses LPAP, deren Startzeitpunkt noch nicht erreicht ist.

master Ist der LPAP-Partner Slave eines LU6.1-LPAP-Bündels, wird in *master* der Master-LPAP-Partner des Bündels zurückgegeben.

bundle gibt an, ob der LPAP-Partner zu einem LPAP-Bündel gehört.

'N' Der LPAP-Partner gehört zu keinem LPAP-Bündel.

'M' Der LPAP-Partner ist der Master eines LPAP-Bündels.

'S' Der LPAP-Partner ist ein Slave eines LPAP-Bün

out_queue_ex siehe out_queue.

dead_letter_q gibt an, ob eine Asynchron-Nachricht an den LPAP-Partner in die Dead Letter Queue gesichert werden soll, wenn sie wegen eines permanenten Fehlers nicht gesendet werden konnte.

'Y' Asynchron-Nachrichten an diesen LPAP-Partner, die wegen eines permanenten Fehlers nicht gesendet werden konnten, werden in die Dead Letter Queue gesichert, sofern (bei Message- Komplexen) kein negativer Quittungsauftrag definiert wurde.

'N' Asynchron-Nachrichten an diesen LPAP-Partner, die wegen eines permanenten Fehlers nicht gesendet werden konnten, werden gelöscht.

11.3.1.15 kc_lses_str - LU6.1-Sessions

Für den Objekttyp KC_LSES ist die Datenstruktur *kc_lses_str* definiert. In *kc_lses_str* liefert UTM bei KC_GET_OBJECT die Eigenschaften von Sessions zu LU6.1-Partnern der Anwendung zurück.

Sessions zu LU6.1-Partnern können dynamisch mit KC_CREATE_OBJECT erzeugt, mit KC_DELETE_OBJECT gelöscht oder mit KC_MODIFY_OBJECT modifiziert werden.

Eine Session ist durch den in der LSES-Anweisung angegebenen Namen identifizierbar.

mod[1]	**Datenstruktur kc_lses_str**
-	`char ls_name[8];`
-	`char lpap[8];`
-	`char rses[8];`
-	`char con[8];`
-	`char pronam[8];`
-	`char bcamappl[8];`
x(A)	`char connect_mode;`
x(IR)	`char quiet_connect;`
-	`char lses_user[8];`
-	`char ls_deleted;`
-	`char ls_used;`
-	`char ptc;`
-	`char node_name[8];`
-	`char pronam_long[64];`

[1] Feldinhalt mit KC_MODIFY_OBJECT modifizierbar; siehe im Abschnitt "obj_type=KC_LSES"

Die Felder der Datenstruktur haben die folgende Bedeutung:

ls_name (**l**ocale **s**ession **name**)

enthält den Namen der Session innerhalb der lokalen Anwendung (lokaler Half-Session-Name).

lpap gibt an, welcher Partner-Anwendung die Session zugeordnet ist. *lpap* enthält den Namen des LPAP-Partners, über den sich die Partner-Anwendung anschließt

rses (**r**emote **ses**sion name)
enthält den Namen, den die Session in der Partner-Anwendung hat (remote Half-Session-Name).

con, pronam, bcamappl

identifizieren eindeutig die Transportverbindung, die für diese Session aufgebaut ist bzw. werden soll.

con
enthält den Namen der Transportverbindung zur Partner-Anwendung, der beim dynamischen Erzeugen (KC_CREATE_OBJECT Objekttyp KC_CON) oder bei der KDCDEF-Generierung in der CON-Anweisung definiert wurde.

pronam_long
ist der Name des Rechners, auf dem die Partner-Anwendung abläuft.

bcamappl
enthält den Namen der lokalen UTM-Anwendung (BCAMAPPL-Name), über den die Verbindung zur Partner-Anwendung aufgebaut wird.

pronam

Wenn der reale Rechnername länger als 8 Zeichen ist, dann gilt Folgendes:

- Das Feld *pronam* enthält einen symbolischen lokalen Namen, der vom Transportsystem für diesen Rechner vergeben wurde.
- War noch keine Verbindung aufgebaut, dann enthält *pronam* Leerzeichen.

connect_mode

gibt an, ob für die Session eine Transportverbindung aufgebaut ist.

'Y' Für die Session ist eine Transportverbindung zur Partner-Anwendung aufgebaut.

'N' Für die Session existiert zur Zeit keine Transportverbindung.

quiet_connect

gibt an, ob für die Verbindung die Eigenschaft QUIET gesetzt ist. QUIET bedeutet, dass UTM die Verbindung abbaut, sobald die Session nicht mehr durch Aufträge belegt ist. Für die Partner-Anwendung werden keine neuen Dialog-Aufträge mehr angenommen.

'Y' Die Eigenschaft QUIET ist gesetzt.

'N' Die Eigenschaft QUIET ist nicht gesetzt

lses_user

Name des Auftraggebers, für den die Session z.Zt. belegt ist. *lses_user* gibt an, wer den Auftraggeber-Vorgang gestartet hat.
Läuft bei einem Dialog-Auftrag der Auftraggeber-Vorgang in der lokalen Anwendung ab, dann wird für *lses_user* die Benutzerkennung oder der LTERM-Partner des Client angegeben, die/der den Vorgang gestartet hat.

Läuft bei einem Dialog-Auftrag der Auftragnehmer-Vorgang in der lokalen Anwendung ab, d.h. die lokale Anwendung bearbeitet den Auftrag, dann wird für *lses_user* der lokale Session-Name (*ls_name*) ausgegeben.

Werden auf der Session Asynchron-Nachrichten übertragen, dann wird für *lses_user* ebenfalls der lokale Session-Name (*ls_name*) ausgegeben.

ls_deleted

zeigt an, ob das LSES-Objekt dynamisch aus der Konfiguration gelöscht wurde.

'Y' Die Session ist gelöscht.

'N' Die Session ist nicht gelöscht.

ls_used

zeigt an, ob die Session benutzt wird oder nicht.

'Y' Die Session wird benutzt.

'N' Die Session wird nicht benutzt.

ptc zeigt an, in welchem Zustand sich die Session befindet.

'Y' Die Session ist im Zustand PTC (prepare to commit).

'N' Die Session ist nicht im Zustand PTC.

node_name

Nur in UTM-Cluster-Anwendungen: Referenzname der Knoten-Anwendung, dem die Session zugeordnet ist.

pronam_long

ist der Name des Rechners, auf dem die Partner-Anwendung abläuft. Die Namen in den Feldern con, pronam_long und bcamappl identifizieren eindeutig die Transportverbindung, die für diese Session aufgebaut ist bzw. werden soll.

11.3.1.16 kc_ltac_str - Transaktionscodes ferner Services (LTAC)

Für den Objekttyp KC_LTAC ist die Datenstruktur *kc_ltac_str* definiert. In *kc_ltac_str* liefert UTM bei KC_GET_OBJECT die Eigenschaften von Transaktionscodes zurück, die in der lokalen Anwendung für ferne Service-Programme definiert sind.

LTAC-Objekte können dynamisch mit KC_CREATE_OBJECT erzeugt, mit KC_DELETE_OBJECT gelöscht oder mit KC_MODIFY_OBJECT modifiziert werden.

mod [1]	**Datenstruktur kc_ltac_str**
-	`char lc_name[8];`
-	`char lpap[8];`
-	`union kc_rtac rtac;`
-	`char rtac_lth[2];`
-	`char code_type;`
x(GPR)	`char state;`
x(GPR)	`char accesswait_sec[5];`
x(GPR)	`char replywait_sec[5];`
-	`char lock_code[4];`
-	`char ltac_type;`
-	`char ltacunit[4];`
-	`char used[10];`
-	`char access_list[8];`
-	`char deleted;`

[1] Feldinhalt mit KC_MODIFY_OBJECT modifizierbar; siehe im Abschnitt "obj_type=KC_LTAC"

Die Felder der Datenstruktur haben die folgende Bedeutung:

lc_name

enthält den lokalen Transaktionscode, der für das ferne Service-Programm definiert wurde (LTAC-Name).

lpap

gibt an, zu welcher Partner-Anwendung das Service-Programm gehört. *lpap* enthält den Namen des LPAP- oder OSI-LPAP-Partners, der der Partner-Anwendung zugeordnet ist, oder den Namen eines Master-LPAP-Partners.

Enthält *lpap* Leerzeichen, dann ist der LTAC keiner Partner-Anwendung explizit zugeordnet. Eine Zuordnung muss dann im KDCS-Aufruf APRO erfolgen.

rtac

(**r**emote **tac**)
Name des Transaktionscodes für den Vorgang bzw. das Service-Programm in der Partner-Anwendung (recipient TPSU-title; RTAC-Name). Der Name kann ein String oder eine Zahl sein. Der Name wird in der folgenden Struktur zurückgeliefert:

union kc_rtac
`char name64[64];`
`char name8[8];`

In welchem Union-Feld der RTAC-Name steht, ist abhängig vom Codetyp, der dem RTAC-Namen zugeordnet ist. Der Codetyp wird im Feld *code_type* zurückgeliefert.

rtac_lth

gibt an, wie lang der in *rtac* zurückgelieferte Name (recipient TPSU-title) ist. Angegeben wird die Anzahl der relevanten Byte in *rtac*.

Minimalwert: '1'
Maximalwert: '64'

code_type

gibt an, welcher Codetyp von UTM intern für den RTAC-Namen verwendet wird. Anhand von *code_type* können Sie ablesen, in welchem Union-Feld der RTAC-Name abgelegt ist.

'I' (INTEGER)
Der TAC-Name in *rtac* ist eine ganze positive Zahl zwischen 0 und 67108863.

Der RTAC-Name wird im Feld *name8* der Union *kc_rtac* zurückgeliefert (die ersten 8 Byte der Union sind rechtsbündig belegt).

RTAC-Namen vom Codetyp INTEGER sind nur für Partner-Anwendungen zulässig, die keine UTM-Anwendungen sind und über das OSI TP-Protokoll kommunizieren.

'P' (PRINTABLE-STRING)
Der TAC-Name in *rtac* ist als String angegeben. Er ist max. 64 Zeichen lang.
Klein-/Großschreibung ist relevant.

Ein TAC-Name mit Codetyp PRINTABLE-STRING kann folgende Zeichen enthalten:

- A, B, C, . . . , Z
- a, b, c, . . . , z
- 0, 1, 2, . . . , 9
- die Sonderzeichen ' - : ? = , + . ( ) / 'BLANK' (Leerzeichen)

'T'

Der TAC-Name wird im Feld *name64* der Union *kc_rtac* zurückgeliefert. Nach der in *rtac_lth* angegebenen Länge ist *kc_rtac.name64* mit Leerzeichen aufgefüllt.T61-STRING *rtac* enthält einen T61-String. Für den Codetyp T61-STRING unterstützt UTM alle Zeichen des Codetyps PRINTABLE-STRING und zusätzlich folgende Sonderzeichen: $ > < & @ # % ; * _Der TAC-Name wird im Feld *name64* der Union *kc_rtac* zurückgeliefert. Nach der in *rtac_lth* angegebenen Länge ist *rtac.name64* mit Leerzeichen aufgefüllt.

RTAC-Namen, für die beim dynamischen Erzeugen oder bei der KDCDEF-Generierung der Codetyp 'S' bzw. STANDARD angegeben wurde, werden je nach verwendeten Zeichen intern als PRINTABLE-STRING bzw. als T61-STRING abgelegt. Daher wird hier für RTACs, die mit 'S' bzw. STANDARD erzeugt wurden, entweder PRINTABLE-STRING oder T61-STRING ausgegeben.

state enthält den Status des Transaktionscodes in *lc_name*:

'Y' Der Transaktionscode ist nicht gesperrt. Aufträge für den zugehörigen fernen Service werden angenommen.

'N' Der Transaktionscode ist gesperrt. Aufträge für den zugehörigen fernen Service werden nicht angenommen.

accesswait_sec

Zeit in Sekunden, die nach dem Anfordern des fernen Services (Aufruf des LTACs) auf das Belegen einer Session (evtl. einschließlich Verbindungsaufbau) bzw. auf den Aufbau einer Association maximal gewartet wird.

Eine Wartezeit *accesswait_sec* != 0 bedeutet bei Asynchron-Aufträgen (LTAC mit *ltac_type*='A'), dass der Auftrag immer in die lokale Message Queue für die Partner-Anwendung eingetragen wird. Dialog-Aufträge werden angenommen.

Die Wartezeit *accesswait_sec*=0 bedeutet:
Dialog-Aufträge werden abgewiesen, wenn zu dem Partner keine Session/Association generiert ist, für die die lokale Anwendung Contention Winner ist.
Bei Asynchron-Aufträgen wird der FPUT-Aufruf mit einem Returncode abgewiesen, falls zur Partner-Anwendung keine logische Verbindung besteht. Besteht eine logische Verbindung zur Partner-Anwendung, dann wird die Nachricht in die lokale Message Queue eingetragen.

Dialog-Aufträge werden unabhängig vom Wert in *accesswait_sec* abgewiesen, wenn keine logische Verbindung zur Partner-Anwendung besteht. Gleichzeitig wird ein Verbindungsaufbau angestoßen.

Minimalwert: '0'

Maximalwert: '32767'

replywait_sec

Zeit in Sekunden, die UTM maximal auf die Antwort vom fernen Service wartet. Durch Begrenzung der Wartezeit kann gewährleistet werden, dass die Wartezeit für Clients bzw. Benutzer am Terminal nicht beliebig lang werden kann.

replywait_sec='0' bedeutet: Warten ohne Zeitbegrenzung.

Minimalwert: '0'
Maximalwert: '32767'

lock_code

enthält den Lockcode, der dem fernen Service innerhalb der lokalen Anwendung zugeordnet ist (Zugriffsschutz). *lock_code* kann eine Zahl zwischen '0' und '4000' enthalten. In KC_CREATE_OBJECT darf *lock_code* höchstens den mit dem Operanden KEYVALUE der KDCDEF-Anweisung MAX definierten Maximalwert enthalten. '0' bedeutet, dass der LTAC nicht durch einen Lockcode geschützt ist.

Beim Erzeugen eines LTAC-Objekts darf nur *lock_code* oder *access_list* spezifiziert werden (siehe unten). Wenn Sie ein LTAC-Objekt modifizieren, können Sie den aktuellen Wert ändern oder den Lockcode entfernen, indem Sie '0' angeben.

Ist weder *lock_code* noch *access_list* definiert, ist *lc_name* nicht geschützt und jeder Benutzer der lokalen UTM-Anwendung kann das ferne Service-Programm starten.

ltac_type

gibt an, ob die lokale Anwendung mit dem fernen Service Aufträge im Dialog bearbeitet oder ob Asynchron-Aufträge an den Partner-Service übergeben werden.

'D' Aufträge an den Partner-Service werden im Dialog bearbeitet.

'A' Der Partner-Service wird asynchron (über Message Queuing) gestartet.

used

enthält die Anzahl der Aufträge, die seit dem Start der lokalen Anwendung an den fernen Service gestellt wurden. *used* gibt also an, wie oft der LTAC innerhalb des aktuellen Anwendungslaufs aufgerufen wurde.

Der Zähler wird bei jedem Start der Anwendung auf 0 zurückgesetzt.

ltacunit

enthält die Anzahl der Verrechnungseinheiten, die in der Abrechnungsphase des UTM-Accounting für jeden Aufruf von *ltac* berechnet wird.
Die Verrechnungseinheiten werden auf den Verrechnungseinheitenzähler der Benutzerkennung aufaddiert, die den *ltac* aufgerufen hat.

Zum Accounting siehe auch openUTM-Handbuch „Anwendungen generieren“ und openUTM-Handbuch „Einsatz von UTM-Anwendungen“.

access_list

kann den Namen eines Keysets enthalten, das die Zugriffsrechte von Benutzern beschreibt, die für *lc_name* zulässig sind.

Beim Erzeugen eines LTAC-Objekts darf nur *lock_code* oder *access_list* spezifiziert werden (siehe oben). Wenn Sie ein LTAC-Objekt modifizieren, können Sie den aktuellen Eintrag ändern oder das Keyset entfernen, indem Sie 8 Leerzeichen angeben.

Ist weder *lock_code* noch *access_list* definiert, ist *lc_name* nicht geschützt und jeder Benutzer der lokalen UTM-Anwendung kann das ferne Service-Programm starten.

deleted

zeigt an, ob *lc_name* dynamisch aus der Konfiguration gelöscht wurde.

'Y' *lc_name* ist gelöscht.

'N' *lc_name* ist nicht gelöscht.

In welchem Union-Feld der RTAC-Name steht, ist abhängig vom Codetyp, der dem RTAC-Namen zugeordnet ist. Der Codetyp wird im Feld *code_type* zurückgeliefert.

11.3.1.17 kc_lterm_str - LTERM-Partner

Für den Objekttyp KC_LTERM ist die Datenstruktur *kc_lterm_str* definiert.

Bei KC_GET_OBJECT liefert UTM in *kc_lterm_str* die Eigenschaften von LTERM-Partnern zurück und gibt an, welcher Client oder Drucker dem LTERM-Partner derzeit zugeordnet ist.

LTERM-Partner können dynamisch mit KC_CREATE_OBJECT erzeugt, mit KC_DELETE_OBJECT gelöscht oder mit KC_MODIFY_OBJECT modifiziert werden.

mod[1]	**Datenstruktur kc_lterm_str**
-	`char lt_name[8];`
-	`char kset[8];`
-	`char locale_lang_id[2];` (nur auf BS2000-Systemen)
-	`char locale_terr_id[2];` (nur auf BS2000-Systemen)
-	`char locale_ccsname[8];` (nur auf BS2000-Systemen)
-	`char lock_code[4];`
x(GPD)	`char state;`
-	`char usage_type;`
-	`char user_gen[8];`
-	`char cterm[8];`
x(GPD)[2]	`char format_attr;` (nur auf BS2000-Systemen)
x(GPD)[2]	`char format_name[7];` (nur auf BS2000-Systemen)
-	`char plev[5];`
-	`char qamsg;`
-	`char qlev[5];`
-	`char restart;`
-	`char annoamsg;` (nur auf BS2000-Systemen)
-	`char netprio;` (nur auf BS2000-Systemen)
x(PD)	`char master[8];`
-	`char pterm[8];`
-	`char pronam[8];`

mod[1]	Datenstruktur kc_lterm_str
-	`char bcamappl[8];`
-	`char user_curr[8];`
x(A)	`char connect_mode;`
x(IR)	`char bcam_trace;`
-	`char bundle;`
-	`char pool;`
-	`char out_queue[5];`
-	`char incounter[10];`
-	`char seccounter[5];`
-	`char deleted;`
-	`char nbr_dputs[10];`
-	`char lt_group;`
-	`char out_queue_ex[10];`
-	`char kerberos_dialog;` (nur auf BS2000-Systemen)
-	`char pronam_long[64];`

[1] Feldinhalt mit KC_MODIFY_OBJECT modifizierbar; siehe Abschnitt "obj_type=KC_LTERM"

[2] Beim Ändern des Startformats mit KC_MODIFY_OBJECT müssen Sie immer *format_name* und *format_attr* belegen.

Die Felder der Datenstruktur haben die folgende Bedeutung:

lt_name

Name des LTERM-Partners; das kann auch ein LTERM-Partner sein, der zu einem LTERM-Pool gehört.

Mit diesem Namen sprechen die Teilprogramme der Anwendung die Clients, Drucker und TS-Anwendungen (keine Server-Server-Kommunikation) an, die dem LTERM-Partner zugeordnet sind.

kset gibt an, welches Keyset diesem LTERM-Partner zugeordnet ist (Zugriffsrechte). *kset* enthält den Namen des Keysets.

Das Keyset schränkt die Zugriffsrechte eines Clients ein, der sich über diesen LTERM-Partner anschließt. Ein Client kann einen mit Lockcode bzw. Access-List gesicherten Service nur starten, wenn sowohl im Keyset der Benutzerkennung, unter der sich der Client anmeldet, als auch im Keyset des zugehörigen LTERM-Partners der zum Lockcode bzw. der Access-List passende Key- bzw. Zugangscode enthalten ist.

locale_lang_id, locale_terr_id, locale_ccsname (nur auf BS2000-Systemen)

enthalten die drei Komponenten des Locale, das dem LTERM-Partner zugeordnet ist. Das Locale definiert die Sprachumgebung des Clients, der sich über diesen LTERM-Partner an die Anwendung anschließt. Die Sprachumgebung ist relevant, wenn Nachrichten und Meldungen der Anwendung in verschiedenen Sprachen ausgegeben werden. Die LTERM-spezifische Sprachumgebung wird bei der Ausgabe von Asynchron-Nachrichten und im ersten Teil des Anmelde-Vorgangs eingestellt, wenn die Benutzer-spezifische Sprachumgebung nicht eingestellt ist.

Zur Mehrsprachigkeit siehe openUTM-Handbuch „Anwendungen generieren“.

locale_lang_id
enthält das bis zu 2 Zeichen lange Sprachkennzeichen.

locale_terr_id
enthält ein bis zu 2 Zeichen langes Territorialkennzeichen.

locale_ccsname
(**c**oded **c**haracter **s**et **name**)
enthält den bis zu 8 Zeichen langen Namen eines erweiterten Zeichensatzes (CCS-Name; siehe auch Benutzerhandbuch zu XHCS).

lock_code

enthält den Lockcode, der dem LTERM-Partner zugeordnet ist (Zugriffsschutz). Nur Benutzer/Clients, die den entsprechenden Keycode besitzen, dürfen sich über diesen LTERM-Partner anschließen.

lock_code kann eine Zahl zwischen '0' und '4000' enthalten. In KC_CREATE_OBJECT darf *lock_code* höchstens den mit dem Operanden KEYVALUE der KDCDEF-Anweisung MAX definierten Maximalwert enthalten.

'0' bedeutet, dass der LTERM-Partner nicht durch einen Lockcode geschützt ist.

state

gibt an, ob der LTERM-Partner derzeit gesperrt ist.

'Y' Der LTERM-Partner ist nicht gesperrt.

'N' Der LTERM-Partner ist gesperrt. Zur Zeit kann sich kein Benutzer/Client über diesen LTERM-Partner an die Anwendung anschließen.

usage_type

Art des LTERM-Partners

'D' Der LTERM-Partner ist als Dialog-Partner konfiguriert. Auf den Verbindungen zwischen dem Client (*pterm*) und lokaler Anwendung kann sowohl der Client als auch die lokale Anwendung Nachrichten senden.

'O' Der LTERM-Partner ist für ein Ausgabemedium (Drucker) konfiguriert. Es können nur Nachrichten von der Anwendung zum Client/Drucker (*pterm*) gesendet werden.

user_gen

ist belegt, wenn der LTERM-Partner als Dialog-Partner konfiguriert ist (*usage_type*='D').

Ist der LTERM-Partner mit einem Terminal zugeordnet, dann enthält *user_gen* die Benutzerkennung, für die UTM beim Aufbau der logischen Verbindung zwischen Client und Anwendung ein automatisches KDCSIGN (automatische Anmeldung) durchführen soll (definiert in *kc_lterm_str.user*).
Ist der LTERM-Partner einem UPIC-Client oder einer TS-Anwendung (*ptype*='APPLI' oder 'SOCKET') zugeordnet, dann enthält *user_gen* die Verbindungs-Benutzerkennung.

cterm

ist nur belegt, wenn der LTERM-Partner (*usage_type*='O') einem Druckersteuer-LTERM zugeordnet ist. *cterm* enthält dann den max. 8 Zeichen langen Namen des Druckersteuer-LTERMs.

Einem Druckersteuer-LTERM werden ein oder mehrere Drucker zugeordnet. Über das Druckersteuer-LTERM können Asynchron-Aufträge in den Message Queues der Drucker, die Ausgabe von Nachrichten auf den Druckern und die Drucker selbst administriert werden (siehe Kapitel „Message Queues administrieren, Druckausgabe steuern").

format_attr, format_name (nur auf BS2000-Systemen)

sind nur relevant, wenn der LTERM-Partner einem Terminal zugeordnet ist.

format_attr und *format_name* definieren das LTERM-spezifische Startformat. Ein LTERM-spezifisches Startformat ist nur nutzbar in Anwendungen ohne Benutzerkennungen oder wenn ein eigener Anmelde-Vorgang definiert ist.

In Anwendungen ohne Benutzerkennungen wird das Startformat nach dem Aufbau der Verbindung zur Anwendung anstelle der Meldung K001 am Terminal ausgegeben, sofern kein LTERM-spezifischer Wiederanlauf durchgeführt wird.

In Anwendungen, die mit Benutzerkennungen generiert sind, kann der Name des Startformats im ersten Teil des Anmelde-Vorgangs abgefragt werden (mit SIGN ST).

format_attr
enthält das Formatkennzeichen:

'A' (Formatattribut ATTR)
Das Startformat ist ein Format mit Benutzerattributen. Die Eigenschaften der Formatfelder können von KDCS-Teilprogrammen verändert werden.
Der Formatname an der Programmschnittstelle KDCS ist +*format_name*.

'N' (Formatattribut NOATTR)
Das Startformat ist ein Format ohne Benutzerattribute. Weder Feld- noch Formateigenschaften können von KDCS-Teilprogrammen verändert werden. Der Formatname an der Programmschnittstelle KDCS ist **format_name*.

'E' (Formatattribut EXTEND)
Das Startformat ist ein Format mit erweiterten Benutzerattributen. Es können sowohl die Eigenschaften der Formatfelder als auch globale Formateigenschaften von KDCS-Teilprogrammen verändert werden. Der Formatname an der Programmschnittstelle KDCS ist #*format_name*.

format_name
enthält den Namen des Startformats. Der Name kann bis zu 7 Zeichen lang sein und enthält nur alphanumerische Zeichen.

plev (**p**rint **lev**el)

Ist der LTERM-Partner ein Dialog-Partner, dann wird immer *plev*='0' zurückgeliefert.

Ist der LTERM-Partner einem Ausgabemedium (Drucker) zugeordnet, dann enthält *plev* den Schwellwert für die Anzahl der Druckaufträge, die in der Message Queue des LTERM-Partners zwischengespeichert sind. UTM sammelt die Nachrichten für die zugehörigen Drucker so lange, bis der in *plev* angegebene Schwellwert erreicht ist. Dann versucht UTM die Verbindung zu den Druckern aufzubauen. Die Verbindung wird wieder abgebaut, wenn keine Nachrichten für die Drucker mehr in der Queue stehen. Der Schwellwert wird beim Eintragen des LTERM-Partners in die Konfiguration festgelegt.

plev='0' heißt, dass kein Schwellwert definiert ist und UTM beliebig viele Druckaufträge in der Queue zwischenspeichert, ohne die Verbindung zu den Druckern aufzubauen.

qamsg (**q**ueue **a**synchronous **m**e**s**sa**g**e)

gibt an, ob Asynchron-Aufträge in der Message Queue des LTERM-Partners zwischengespeichert werden, auch wenn der zum LTERM-Partner gehörende Client/Drucker nicht mit der Anwendung verbunden ist.

'Y' Ein Asynchron-Auftrag wird in die Message Queue des LTERM-Partners übernommen, auch wenn keine Verbindung zum Client/Drucker existiert. *qamsg*='Y' ist nicht möglich bei *restart*='N'.

'N' Ein Asynchron-Auftrag an diesen LTERM-Partner wird abgewiesen (Returncodes KCRCCC=44Z und KCRCDC=K705), falls der zugehörige Client/Drucker nicht mit der Anwendung verbunden ist.

qlev (**q**ueue **lev**el)

enthält die Anzahl der Asynchron-Nachrichten, die UTM in der Message Queue des LTERM-Partners maximal gleichzeitig zwischenspeichert. Wird dieser Schwellwert überschritten, so weist openUTM weitere FPUT- oder DPUT-Aufrufe für diesen LTERM-Partner mit 40Z ab. Der Schwellwert wird beim Eintragen des LTERM-Partners in die Konfiguration festgelegt.

restart

ist nur relevant, wenn der LTERM-Partner einem Client zugeordnet ist. *restart* gibt an, wie UTM beim Abbau einer Verbindung zum Client mit Asynchron-Nachrichten verfährt, die zu diesem Zeitpunkt in der Message Queue des LTERM-Partners stehen.

'Y' Beim Verbindungsabbau bleiben Asynchron-Nachrichten an diesen Client erhalten. Sind in dieser Anwendung keine Benutzerkennungen (USER) generiert, dann führt UTM für diesen LTERM-Partner den automatischen Vorgangswiederanlauf durch.

'N' Beim Verbindungsabbau löscht UTM alle Asynchron-Nachrichten, die zu diesem Zeitpunkt in der Message Queue des LTERM-Partners zwischengespeichert sind.
Handelt es sich dabei um Auftrags-Komplexe, so wird der negative Quittungs-Auftrag aktiviert.

Sind in dieser Anwendung keine Benutzerkennungen (USER) definiert, dann führt UTM für den LTERM-Partner keinen automatischen Vorgangswiederanlauf durch.

annoamsg (**anno**unce **a**synchronous **m**e**s**sa**g**e, nur auf BS2000-Systemen)

ist nur bei LTERM-Partnern relevant, die einem Terminal zugeordnet sind. *annoamsg* gibt an, ob UTM Asynchron-Nachrichten vor der Ausgabe am Terminal durch eine Meldung in der Systemzeile ankündigt.

'Y' UTM kündigt jede Asynchron-Nachricht an dieses Terminal mit der Meldung K012 in der Systemzeile an. Der Benutzer muss die Asynchron-Nachricht dann explizit mit dem Kommando KDCOUT anfordern.

'N' Asynchron-Nachrichten an das Terminal werden sofort, d.h. ohne Ankündigung ausgegeben. KDCOUT ist nicht erlaubt.

netprio gibt die Transportpriorität an, die auf Transportverbindungen zwischen der Anwendung und dem Client /Drucker benutzt wird.

'M' Transportpriorität „Medium“

'L' Transportpriorität „Low“

master

Dieses Feld hat, abhängig vom Operationscode, eine unterschiedliche Bedeutung. Um welche Art von LTERM es sich handelt, können Sie dem Parameter *lt_group* entnehmen.

KC_GET_OBJECT

- Bei einem Slave-LTERM eines LTERM-Bündels wird hier das zugehörige Master-LTERM zurückgegeben.
- Bei einem Alias-LTERM einer LTERM-Gruppe wird hier das zugehörige Primary-LTERM zurückgegeben.

KC_MODIFY_OBJECT

- Bei Verbindungsbündeln: Austausch zweier Master-LTERMs. Das in *master* angegebene LTERM muss Master eines LTERM-Bündels sein. Es wird der Master angegeben, mit dem die Slaves getauscht werden sollen.

  Diese Funktionalität steht Ihnen nur in stand-alone UTM-Anwendungen zur Verfügung.
- Bei LTERM-Gruppen: Umhängen eines Gruppen-LTERMs in eine andere LTERM-Gruppe. Das bei *master* angegebene LTERM muss ein normales LTERM, ein Primary-LTERM einer LTERM-Gruppe oder das Master-LTERM eines LTERM-Bündels sein.

  Diese Funktionalität steht Ihnen nur in stand-alone UTM-Anwendungen zur Verfügung.

  Ein normales LTERM muss dabei folgende Bedingungen erfüllen:
  - Dem LTERM muss ein PTERM mit PTYPE APPLI oder SOCKET zugeordnet sein.
  - Das LTERM darf kein Slave-LTERM eines LTERM-Bündels sein.
  - Das LTERM muss mit USAGE=D generiert sein.

Es wird das Primary-LTERM angegeben, in dessen Gruppe das LTERM aufgenommen werden soll.
Ist das bei *master* angegebene LTERM bereits ein Primary-LTERM einer LTERM-Gruppe, so wird das bei *lt_name* angegebene LTERM in dessen LTERM-Gruppe aufgenommen.
War das bei *master* angegebene LTERM noch kein Primary-LTERM, wird eine neue LTERM-Gruppe angelegt. Das bei *lt_name* angegebene LTERM wird in die neue LTERM-Gruppe aufgenommen. Primary-LTERM ist das bei *master* angegebenen LTERM.

pterm

Name des Client/Druckers (PTERM-Name), der diesem LTERM-Partner zur Zeit zugeordnet ist. Ist dem LTERM-Partner zur Zeit kein Client/Drucker zugeordnet, dann ist *pterm* mit Leerzeichen belegt. Die Zuordnung zwischen LTERM-Partner und Client/Drucker kann geändert werden, siehe dazu *kc_pterm_str* im Abschnitt "kc_pterm_str - Clients und Drucker".

pronam

Name des Rechners, an/auf dem sich der Client/Drucker befindet.

Wenn der reale Rechnername länger als 8 Zeichen ist, dann gilt Folgendes:

- Das Feld *pronam* enthält einen symbolischen lokalen Namen, der vom Transportsystem für diesen Rechner vergeben wurde.
- Der vollständige, bis zu 64 Zeichen lange Name kann dem Feld *pronam_long* entnommen werden.
- War noch keine Verbindung aufgebaut, dann enthält *pronam* Leerzeichen.

Ist dem LTERM-Partner zur Zeit kein Client/Drucker zugeordnet, ist das Feld mit Leerzeichen belegt.

In UTM-Anwendungen auf BS2000-Systemen ist *pronam* ungleich Leerzeichen, wenn dem LTERM-Partner ein Client oder Drucker zugeordnet ist. Der Name in *pronam* ist identisch mit dem Rechnernamen, der bei der BCAM-Generierung für diesen Rechner festgelegt wurde. Ist der LTERM-Partner einem RSO-Drucker zugeordnet, dann enthält *pronam* den Wert '*RSO'.

In UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen enthält *pronam* Leerzeichen, wenn der LTERM-Partner einem lokalen Client bzw. Drucker zugeordnet ist.

bcamappl

Name der lokalen UTM-Anwendung (BCAMAPPL-Name), über den die Verbindung zu dem Client /Drucker aufgebaut wird.

Ist der LTERM-Partner einem Terminal oder Drucker zugeordnet, dann enthält *bcamappl* immer den Namen der Anwendung, der bei der KDCDEF-Generierung in MAX APPLINAME festgelegt wurde. Ist dem LTERM-Partner ein UPIC-Client oder eine TS-Anwendung zugeordnet, dann enthält *bcamappl* den Anwendungsnamen (BCAMAPPL-Name), der dem Client beim Eintragen zugeordnet wurde.

user_curr

Benutzerkennung, die derzeit über diesen LTERM-Partner mit der Anwendung verbunden ist. Besteht derzeit keine Verbindung, dann ist *user_curr* mit Leerzeichen belegt.

Ist eine Verbindung zu einem Terminal aufgebaut, aber noch kein Benutzer angemeldet, dann ist *user_curr* ebenfalls mit Leerzeichen belegt.

Ist eine Verbindung zu einem UPIC-Client oder einer TS-Anwendung aufgebaut, sind folgende Fälle zu unterscheiden:

- Die Anwendung ist mit SIGNON MULTI-SIGNON=YES generiert (siehe *kc_signon_str.multi_signon* im Abschnitt "kc_signon_str - Eigenschaften des Anmeldeverfahrens"):
  user_curr enthält die Verbindungs-Benutzerkennung (*user_gen*), solange bis sich ein Client mit einer „echten“ Benutzerkennung anmeldet, für die *kc_user_str.restart*='Y' gesetzt ist.
- Die Anwendung ist mit SIGNON MULTI-SIGNON=NO generiert:*user_curr* enthält die Verbindungs-Benutzerkennung (*user_gen*), solange bis sich ein Client mit einer „echten“ Benutzerkennung anmeldet.

connect_mode

gibt an, ob der Client bzw. Drucker, dem dieser LTERM-Partner aktuell zugeordnet ist, zur Zeit mit der Anwendung verbunden ist.

'Y' Der Client/Drucker ist zur Zeit mit der Anwendung verbunden.

'W' UTM versucht gerade eine Verbindung zu dem Client/Drucker aufzubauen.

'N' Der Client/Drucker ist zur Zeit nicht mit der Anwendung verbunden.

bcam_trace

gibt an, ob der BCAM-Trace explizit für diesen LTERM-Partner eingeschaltet ist oder nicht. Als BCAM-Trace wird die Tracefunktion bezeichnet, die Verbindungsspezifische Aktivitäten innerhalb einer UTM-Anwendung verfolgt (z.B. BCAM-Tracefunktion auf BS2000-Systemen). Der BCAM-Trace kann allgemein für alle Verbindungen der Anwendung (d.h. für alle LPAP- und LTERM-Partner) eingeschaltet werden oder explizit für bestimmte LPAP- oder LTERM-Partner.

'Y' Der BCAM-Trace ist explizit für diesen LTERM-Partner eingeschaltet.
Ist der BCAM-Trace allgemein für alle Verbindungen der UTM-Anwendung eingeschaltet, dann wird in *bcam_trace* 'N' zurückgeliefert.

Ob der BCAM-Trace allgemein eingeschaltet ist, können Sie z.B. durch einen KC_GET_OBJECT-Aufruf mit Parametertyp KC_DIAG_AND_ACCOUNT_PAR ermitteln. Es wird dann *bcam_trace* ='Y' in *kc_diag_and_account_par_str* zurückgeliefert.

'N' Der BCAM-Trace ist nicht explizit für diesen LTERM-Partner eingeschaltet worden.

Der BCAM-Trace kann während des Anwendungslaufs ein- und ausgeschaltet werden.

bundle ist nur bei LTERM-Partnern relevant, die einem Drucker oder einem LTERM-Bündel zugeordnet sind. bundle gibt an, ob der LTERM-Partner zu einem Drucker- oder LTERM-Bündel gehört.

'Y' Der Drucker ist einem Druckerbündel zugeordnet.

'N' Der Drucker ist keinem Druckerbündel zugeordnet.

'M' Der LTERM-Partner ist ein Master eines LTERM-Bündels.

'S' Der LTERM-Partner ist ein Slave eines LTERM-Bündels.

pool gibt an, ob der LTERM-Partner zu einem LTERM-Pool gehört.

'Y' Der LTERM-Partner ist einem LTERM-Pool zugeordnet.

'N' Der LTERM-Partner ist keinem LTERM-Pool zugeordnet.

out_queue

Anzahl der Asynchron-Nachrichten, die zur Zeit in der Message Queue des LTERM-Partners zur Ausgabe anstehen.

Ist die Anzahl der Nachrichten größer als 99999, wird die Zahl nur unvollständig dargestellt. Deshalb sollte das Feld *out_queue_ex* verwendet werden, weil hier auch größere Zahlen vollständig abgebildet werden.

incounter

Anzahl der Nachrichten, die über diesen LTERM-Partner eingegeben wurden; ist über den LTERM-Partner ein Drucker angeschlossen, dann wird hier die Anzahl der Abdruckquittungen vom Drucker eingetragen.
Der Zähler *incounter* wird bei jedem Start der Anwendung auf 0 zurückgesetzt.

seccounter

Anzahl der Sicherheitsverletzungen von Benutzern und Clients, die über diesen LTERM-Partner mit der Anwendung verbunden waren (z.B. nicht erlaubter Transaktionscode eingegeben).
Der Zähler wird bei jedem Start der Anwendung auf 0 zurückgesetzt.

deleted

gibt an, ob der LTERM-Partner dynamisch aus der Konfiguration gelöscht wurde oder nicht.

'Y' Der LTERM-Partner wurde gelöscht. Über ihn können keine Clients oder Drucker mehr mit der Anwendung verbunden werden.

'N' Der LTERM-Partner wurde gelöscht. Über ihn können keine Clients oder Drucker mehr mit der Anwendung verbunden werden.

nbr_dputs

Anzahl zeitgesteuerter Aufträge für diesen LTERM-Partner, deren Startzeitpunkt noch nicht erreicht ist.

lt_group

gibt an, ob das LTERM ein „normales“ LTERM, Teil eines LTERM-Bündels oder Teil einer LTERM-Gruppe ist.

' ' Das LTERM ist nicht Teil eines LTERM-Bündels oder einer LTERM-Gruppe.

'P' Das LTERM ist das Primary-LTERM einer LTERM-Gruppe.

'A' Das LTERM ist ein Alias-LTERM in einer LTERM-Gruppe.

out_queue_ex

siehe *out_queue.*

kerberos_dialog (nur auf BS2000-Systemen)

Y Beim Verbindungsaufbau wird für Clients, die Kerberos unterstützen und die über diesen LTERM-Partner direkt (nicht über OMNIS) mit der Anwendung verbunden sind, ein Kerberos-Dialog durchgeführt.

N Es wird kein Kerberos-Dialog durchgeführt.

pronam_long

Name des Rechners, auf dem sich der Client bzw. an dem sich der Drucker befindet. Ist dem LTERM-Partner zur Zeit kein Client/Drucker zugeordnet, ist das Feld mit Leerzeichen belegt. In UTM-Anwendungen auf BS2000-Systemen ist pronam_long ungleich Leerzeichen, wenn dem LTERM-Partner ein Client oder Drucker zugeordnet ist. Der Name in pronam_long ist identisch mit dem

Rechnernamen, der bei der BCAMGenerierung für diesen Rechner festgelegt wurde. Ist der LTERM-Partner einem RSO-Drucker zugeordnet, dann enthält pronam_long den Wert '*RSO'. In UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen enthält pronam_long Leerzeichen, wenn der LTERM-Partner einem lokalen Client bzw. Drucker zugeordnet ist.

11.3.1.18 kc_message_module_str - Benutzereigene Meldungsmodule

Für den Objekttyp KC_MESSAGE_MODULE ist die Datenstruktur *kc_message_module_str* definiert. In *kc_message_module_str* liefert UTM bei KC_GET_OBJECT die Eigenschaften der benutzereigenen Meldungsmodule der Anwendung zurück.

In UTM-Anwendungen auf BS2000-Systemen können Sie zur Internationalisierung der Anwendung mehrere benutzereigene Meldungsmodule erzeugen, die die UTM-Meldungen in verschiedenen Landessprachen enthalten. Zur genauen Definition der Sprache wird jedem Meldungsmodul ein Sprachkennzeichen und ein Territorialkennzeichen zugeordnet. Die Kombination aus Sprach- und Territorialkennzeichen muss genau einem Meldungsmodul der Anwendung zugeordnet sein. Über das Paar „Sprach- und Territorialkennzeichen“ werden die benutzereigenen Meldungsmodule den Benutzern und LTERM-Partnern zugeordnet, deren Locale dasselbe Sprach- und Territorialkennzeichen beinhalten.

In UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen können Sie ein benutzereigenes Meldungsmodul erstellen. UTM liefert hier nur den Namen des Meldungsmoduls zurück. Die anderen Felder der Datenstruktur haben hier keine Bedeutung.

Benutzer-spezifische Meldungsmodule werden bei der KDCDEF-Generierung mit einer MESSAGE-Anweisung definiert.

Wie ein eigenes Meldungsmodul erzeugt wird, ist im openUTM-Handbuch „Meldungen, Test und Diagnose“ beschrieben.

Datenstruktur kc_message_module_str
`char mm_name[8];`
`char lib[54];` (nur auf BS2000-Systemen)
`char locale_lang_id[2];` (nur auf BS2000-Systemen)
`char locale_terr_id[2];` (nur auf BS2000-Systemen)
`char standard_module;` (nur auf BS2000-Systemen)

Die Felder der Datenstruktur haben die folgende Bedeutung:

mm_name

enthält den Namen des Meldungsmoduls, dessen Eigenschaften UTM zurückliefert.

lib (nur auf BS2000-Systemen)

enthält den Namen der Bibliothek, in der das Meldungsmodul steht.

locale_lang_id, locale_terr_id (nur auf BS2000-Systemen)

geben die Sprachumgebung an, für die das Meldungsmodul verwendet wird.

locale_lang_id
enthält das bis zu 2 Zeichen lange Sprachkennzeichen.

locale_terr_id
enthält ein bis zu 2 Zeichen langes Territorialkennzeichen.

Die Meldungen Benutzer-spezifischer Meldungsmodule werden für die Meldungsziele STATION, SYSLINE und PARTNER verwendet. Es wird jeweils das Meldungsmodul verwendet, für das *locale_lang_id* und *locale_terr_id* mit dem Sprach- und Territorialkennzeichen des Locale vom betroffenen Benutzer bzw. LTERM-Partner identisch sind.

standard_module (nur auf BS2000-Systemen)

gibt an, ob das Meldungsmodul das benutzereigene Standard-Meldungsmodul der Anwendung ist.

Standard-Meldungsmodul ist das benutzereigene Meldungsmodul, dem Sprach- und Territorialkennzeichen der Standard-Sprachumgebung zugeordnet ist. Die Standard-Sprachumgebung wird bei der KDCDEF-Generierung in MAX LOCALE festgelegt.
Das Standard-Meldungsmodul wird von UTM immer für Meldungen an die Meldungsziele SYSLST, SYSOUT und CONSOLE verwendet.

'Y' Das Meldungsmodul ist das Standard-Meldungsmodul.

'N' Das Meldungsmodul ist nicht das Standard-Meldungsmodul.

11.3.1.19 kc_mux_str - Multiplexanschlüsse (BS2000-Systeme)

Für den Objekttyp KC_MUX ist die Datenstruktur *kc_mux_str* definiert. In *kc_mux_str* liefert UTM bei KC_GET_OBJECT Namen und Eigenschaften eines Multiplexanschlusses zurück, über den sich ein Nachrichtenverteiler an die Anwendung anschließen kann.

Über einen Multiplexanschluss können sich mehrere Terminal-Clients gleichzeitig an die UTM-Anwendung anschließen.

mod[1]	**Datenstruktur kc_mux_str**
-	`char mx_name[8];`
-	`char pronam[8];`
-	`char bcamappl[8];`
x(GPD)	`char auto_connect;`
x(GPR)	`char maxses[5];`
x(GPD)	`char state;`
-	`char netprio;`
x(A)	`char connect_mode;`
-	`char actcon[5];`
-	`char maxcon[5];`
-	`char letters[10];`
-	`char incnt[5];`
-	`char wait_go[5];`
-	`char shortage[5];`
-	`char rtryo[5];`
-	`char rtryi[5];`
x(IR)	`char bcam_trace;`

[1] Feldinhalt mit KC_MODIFY_OBJECT modifizierbar; siehe Abschnitt "obj_type=KC_MUX (BS2000-Systeme)"

Die Felder der Datenstruktur haben die folgende Bedeutung:

mx_name

enthält den Namen des Multiplexanschlusses.

pronam

Name des Rechners, auf dem sich der Nachrichtenverteiler befindet. Der Name in *pronam* ist identisch mit dem Rechnernamen, der bei der BCAM-Generierung für diesen Rechner festgelegt wurde.

bcamappl

Name der lokalen UTM-Anwendung (BCAMAPPL-Name), über den die Verbindung zu dem Nachrichtenverteiler aufgebaut wird. Der Nachrichtenverteiler muss beim Verbindungsaufbau zur UTM-Anwendung diesen Anwendungsnamen als Partnernamen angeben.

Existieren in der lokalen Anwendung für einen Nachrichtenverteiler mehrere Multiplexanschlüsse (in *pronam* steht immer derselbe Rechnername) mit verschiedenen BCAMAPPL-Namen, dann können zu dem Nachrichtenverteiler mehrere parallele Verbindungen aufgebaut werden.

auto_connect

gibt an, ob die lokale Anwendung beim Anwendungsstart automatisch eine Transportverbindung zu dem Nachrichtenverteiler aufbaut.

'N' Die Verbindung wird nicht automatisch aufgebaut, sie muss vom Administrator aufgebaut werden (siehe connect_mode).

'Y' Beim Start der lokalen Anwendung versucht UTM die Verbindung zum Nachrichtenverteiler aufzubauen.
Kommt keine Verbindung zustande (z.B. Nachrichtenverteiler ist nicht verfügbar), dann wiederholt UTM den Versuch, die Verbindung aufzubauen und zwar im Zeitabstand, der im Timer *conrtime_min* festgelegt ist. Dieser Timer kann geändert werden (siehe Datenstruktur *kc_timer_par_str* Feld *conrtime_min* im Abschnitt "kc_timer_par_str - Timer-Einstellungen").

maxses

gibt an, wieviele Sessions maximal gleichzeitig zwischen Nachrichtenverteiler und Anwendung bestehen können. *maxses* enthält die maximale Anzahl der Clients, die gleichzeitig über den Nachrichtenverteiler mit der Anwendung verbunden sein können.

Minimalwert:'1'
Maximalwert:'65000' (theoretischer Wert)

state gibt an, ob der Multiplexanschluss derzeit gesperrt ist.

'Y' Der Multiplexanschluss ist nicht gesperrt.

'N' Der Multiplexanschluss ist gesperrt. Zur Zeit kann keine Verbindung zwischen Nachrichtenverteiler und Anwendung aufgebaut werden.

netprio

gibt die Transportpriorität an, die auf Transportverbindungen zwischen der Anwendung und dem Nachrichtenverteiler benutzt wird.

'M' Transportpriorität „Medium“

'L' Transportpriorität „Low“

connect_mode

gibt an, ob der Nachrichtenverteiler zur Zeit mit der Anwendung verbunden ist.

'Y' Der Nachrichtenverteiler ist zur Zeit mit der Anwendung verbunden.

'W' UTM versucht gerade eine Verbindung zu dem Nachrichtenverteiler aufzubauen.

'N' Der Nachrichtenverteiler ist zur Zeit nicht mit der Anwendung verbunden.

actcon

enthält die Anzahl der Clients, die derzeit über diesen Multiplexanschluss mit der Anwendung verbunden sind.

maxcon

enthält den maximalen Wert, den *actcon* im Laufe des aktuellen Anwendungslaufs angenommen hat. *maxcon* gibt also die maximale Anzahl der Clients an, die während des bisherigen Anwendungslaufs gleichzeitig über diesen Multiplexanschluss mit der Anwendung verbunden waren.
Der Zähler wird bei jedem Start der Anwendung auf 0 zurückgesetzt.

letters enthält die Anzahl der Nachrichten, die seit dem Start der Anwendung zwischen Nachrichtenverteiler und Anwendung ausgetauscht wurden (Ein- und Ausgabe-Nachrichten).

incnt enthält die Anzahl der Eingabe-Nachrichten, die von der Anwendung über diesen Multiplexanschluss empfangen wurden.
Der Zähler wird bei jedem Start der Anwendung auf 0 zurückgesetzt.

wait_go

gibt an, wie oft BCAM den Multiplexanschluss auffordern musste, eine Nachricht nochmals zu senden, weil BCAM diese Nachricht zuvor wegen eines BCAM-Engpasses (WAIT FOR GO) nicht annehmen konnte.
Der Zähler wird bei jedem Start der Anwendung auf 0 zurückgesetzt.

shortage

enthält die Anzahl der BCAM-Engpässe (shortages) für diese Multiplexverbindung seit dem Start der Anwendung.

rtryo (**re**t**ry o**ut)

gibt an, wie oft die Anwendung seit dem Anwendungsstart versuchen musste, eine Ausgabe-Nachricht an den Nachrichtenverteiler nochmals zu senden.

rtryi (**re**t**ry i**n)

gibt an, wie oft die Anwendung seit dem Anwendungsstart versuchen musste, eine Nachricht vom Nachrichtenverteiler nochmals zu lesen.
Trifft eine Nachricht vom Nachrichtenverteiler bei BCAM ein, dann teilt BCAM UTM mit, dass eine Nachricht vorliegt. UTM versucht dann die Nachricht von BCAM zu übernehmen. *rtryi* enthält die Anzahl der fehlgeschlagenen Versuche die Nachricht von BCAM zu übernehmen, bevor UTM die Nachricht schließlich lesen konnte.

bcam_trace

gibt an, ob der BCAM-Trace für diesen Multiplexanschluss explizit ein- oder ausgeschaltet ist.

'Y' BCAM-Trace ist explizit eingeschaltet.

'N' BCAM-Trace ist nicht explizit eingeschaltet.

Die Auswertung dieses Feldes ist nur dann sinnvoll, wenn der BCAM-Trace explizit für einige LTERM-Partner, LPAP-Partner bzw. Multiplex-Anschlüsse eingeschaltet ist.

Ist der BCAM-Trace allgemein ein- oder ausgeschaltet (*kc_diag_and_account_par_str*), wird 'N' für *bcam_trace* zurückgegeben.

Soll der Wert von *bcam_trace* modifiziert werden, gelten folgende Voraussetzungen für das explizite Einschalten:

- BCAM-Trace muss für alle ausgeschaltet sein (*kc_diag_and_account_par*) und
- BCAM-Trace muss explizit für diesen Multiplexanschluss ausgeschaltet sein.

Voraussetzung für das explizite Ausschalten ist, dass BCAM-Trace für einige LTERM-Partner bzw. LPAP-Partner bzw. Multiplex-Anschlüsse explizit eingeschaltet ist.

11.3.1.20 kc_osi_association_str - Associations zu OSI TP-Partner-Anwendungen

Für den Objekttyp KC_OSI_ASSOCIATION ist die Datenstruktur *kc_osi_association_str* definiert. In *kc_osi_association_str* liefert UTM bei KC_GET_OBJECT die Eigenschaften einer Association zurück, die zur Zeit für die verteilte Verarbeitung über OSI TP existiert bzw. aufgebaut wird.

Datenstruktur kc_osi_association_str
`char association_id[8];`
`char osi_lpap[8];`
`char contwin;`
`char connect_state;`
`char contime_min[10];`
`char request_calls[10];`
`char indication_calls[10];`

Die Felder der Datenstruktur haben die folgende Bedeutung:

association_id

enthält die Identifikation (Id), die der Association beim Aufbau zugeordnet wurde. Sie ist nur solange eindeutig, wie die Association aufgebaut ist. Wird diese Association abgebaut, dann wird die Id freigegeben und kann einer anderen (danach aufgebauten) Association zugeordnet werden.

Die Association-Id ist eine maximal 8-stellige ganze Zahl.

osi_lpap

gibt die Partner-Anwendung an, zu der die Association aufgebaut ist. In *osi_lpap* liefert UTM den Namen des OSI-LPAP-Partners zurück, dem die Partner-Anwendung zugeordnet ist.

contwin (**cont**ention **win**ner)

gibt an, ob die lokale Anwendung für diese Association der Contention Winner oder der Contention Loser ist.

Der Contention Winner übernimmt die Verwaltung der Association. Aufträge können aber sowohl vom Contention Winner als auch vom Contention Loser gestartet werden. Im Konfliktfall, wenn beide Kommunikationspartner gleichzeitig einen Auftrag starten wollen, wird die Association vom Auftrag des Contention Winner belegt.

'Y' Die lokale Anwendung ist Contention Winner.

'N' Die lokale Anwendung ist Contention Loser.

connect_state

gibt den Status der Association an.

'C' Die Association ist aufgebaut.

'S' Die Association befindet sich in der Aufbauphase im „STOP“-Zustand. Sie wartet auf ein „GO“-Signal von OSS.

contime_min

gibt die Dauer der bestehenden Verbindung in Minuten an.

request_calls

Anzahl der Request-/Response-Presentation-Aufrufe an OSS seit dem Aufbau der Association.

indication_calls

Anzahl der Indication-/Confirmation-Presentation-Aufrufe an OSS seit dem Aufbau der Association.

11.3.1.21 kc_osi_con_str - OSI TP-Verbindungen

Für den Objekttyp KC_OSI_CON ist die Datenstruktur *kc_osi_con_str* definiert. In *kc_osi_con_str* liefert UTM bei KC_GET_OBJECT Namen und Adresse einer OSI TP-Partner-Anwendung und den Zustand der Verbindung zu der Partner-Anwendung zurück.

Eine OSI TP-Verbindung wird mit der KDCDEF-Steueranweisung OSI-CON erzeugt.

mod[1]	**Datenstruktur kc_osi_con_str**
-	`char oc_name[8];`
-	`char osi_lpap[8];`
-	`char local_access_point[8];`
-	`union kc_selector presentation_selector;`
-	`union kc_selector session_selector;`
-	`char presentation_selector_type;`
-	`char presentation_selector_lth[2];`
-	`char presentation_selector_code;`
-	`char session_selector_type;`
-	`char session_selector_lth[2];`
-	`char session_selector_code;`
-	`char transport_selector[8];`
-	`char network_selector[8];`
x(GIR)	`char active;`
-	`char map;` (nur auf Unix-, Linux- und Windows-Systemen)
-	`char listener_port[5];`
-	`char t_prot;` (nur auf Unix-, Linux- und Windows-Systemen)
-	`char tsel_format;` (nur auf Unix-, Linux- und Windows-Systemen)
-	`char ip_addr[15];` (nur auf Unix-, Linux- und Windows-Systemen)
-	`char ip_addr_V6[39];` (nur auf Unix-, Linux- und Windows-Systemen)
-	`char ip_v[2];` (nur auf Unix-, Linux- und Windows-Systemen)
-	`char network_selector_long[64];`

[1] Feldinhalt mit KC_MODIFY_OBJECT modifizierbar; siehe Abschnitt "obj_type=KC_OSI_CON"

Die Felder der Datenstruktur haben die folgende Bedeutung:

oc_name

enthält den Namen einer Verbindung, die mit OSI-CON für die Kommunikation über das OSI TP-Protokoll generiert wurde. *oc_name* identifiziert die Verbindung in der lokalen UTM-Anwendung eindeutig.

osi_lpap

gibt die Partner-Anwendung an, für die die Verbindung definiert ist. *osi_lpap* enthält den Namen des OSI-LPAP-Partners, der der Partner-Anwendung zugeordnet ist.

local_access_point

enthält den Namen eines OSI TP-Zugriffspunktes (Access Point) der lokalen Anwendung (KDCDEF-Anweisung ACCESS-POINT). Über diesen Zugriffspunkt wird die Verbindung zur Partner-Anwendung aufgebaut.

presentation_selector

enthält den Presentation-Selektor der Partner-Anwendung. Der Presentation-Selektor ist eine Komponente der Partneradresse.

presentation_selector ist ein Feld vom Typ *kc_selector*:

union kc_selector
`char x[32];`
`char c[16];`

UTM liefert den Presentation-Selektor i.A. als Character-String (*c*) in Maschinen-spezifischer Codierung (*presentation_selector_code*='S') zurück. Der Character-String ist maximal 16 Zeichen lang. Das Feld *presentation_selector* ist ab der in *presentation_selector_lth* angegebenen Länge mit Leerzeichen aufgefüllt.

In Ausnahmefällen wird der Presentation-Selektor als hexadezimale Zeichenfolge (*x*) zurückgeliefert. Jedes Halb-Byte wird dabei als ein Zeichen dargestellt, z.B. wird die Hexadezimalzahl A2 als String 'A2 ' (2 Zeichen) zurückgeliefert. Ist der Presentation-Selektor eine Hexadezimalzahl, dann liefert UTM bis zu 32 Byte zurück.

Wie der Inhalt von *presentation_selector* zu interpretieren ist, entnehmen Sie dem Feld *presentation_selector_type.*

Enthält die Adresse des Zugriffspunktes keinen Presentation-Selektor, dann ist das Feld *presentation_selector* mit Leerzeichen belegt. In diesem Fall ist *presentation_selector_type*='N' und *presentation_selector_lth*='0'.

session_selector

enthält den Session-Selektor der Partner-Anwendung. Der Session-Selektor ist eine Komponente der Partneradresse.

session_selector ist eine Union vom Typ *kc_selector (*siehe *presentation_selector*).

UTM liefert den Session-Selektor i.A. als Character-String (*c*) in Maschinen-spezifischer Codierung (*session_selector_code*='S') zurück. Der Character-String ist maximal 16 Zeichen lang. Das Feld *session_selector* ist ab der in *session_selector_lth* angegebenen Länge mit Leerzeichen aufgefüllt.

In Ausnahmefällen wird der Session-Selektor als hexadezimale Zeichenfolge (*x*) zurückgeliefert. Jedes Halb-Byte wird dabei als ein Zeichen dargestellt. Ist der Session-Selektor eine Hexadezimalzahl, dann liefert UTM in *session_selector* bis zu 32 Byte zurück.

Wie der Inhalt von *session_selector* zu interpretieren ist, entnehmen Sie dem Feld *session_selector_type.*

Enthält die Adresse des Zugriffspunktes keinen Session-Selektor, dann ist das Feld *session_selector* mit Leerzeichen belegt. In diesem Fall ist *session_selector_type*='N' und *session_selector_lth*='0'.

presentation_selector_type

gibt an, ob die Adresse der Partner-Anwendung einen Presentation-Selektor enthält und wie die Rückgabe in *presentation_selector* zu interpretieren ist.

'N' Steht für *NONE. Die Adresse der Partner-Anwendung enthält keinen Presentation-Selektor, *presentation_selector* ist mit Leerzeichen belegt und *presentation_selector_lth*='0'.

'C' Die Angabe des Presentation-Selektors in *presentation_selector* ist als Character-String zu interpretieren. Es sind maximal die ersten 16 Byte von *presentation_selector* belegt.

'X' Der Presentation-Selektor in *presentation_selector* ist eine Hexadezimalzahl.

presentation_selector_lth

enthält die Länge des Presentation-Selektors (*presentation_selector)* in Byte. Ist *presentation_selector_lth* ='0', dann enthält die Adresse der Partner-Anwendung keine Presentation-Komponente (*presentation_selector* enthält Leerzeichen).
Sonst liegt der Wert von *presentation_selector_lth* zwischen '1' und '16'.

Ist *presentation_selector_type*='X', dann ist die Länge des in *presentation_selector* angegebenen Strings: 2 * *presentation_selector_lth* Byte.

Beispiel:
Der Presentation-Selektor ist X'A2B019CE'. *presentation_selector* enthält dann den String 'A2B019CE', *presentation_selector_type*='X' und *presentation_selector_lth*='4'.

presentation_selector_code

gibt an, wie der Presentation-Selektor in *presentation_selector* codiert ist.

UTM liefert 'S' zurück, wenn der Presentation-Selektor als Character-String zurückgeliefert wird (*presentation_selector_type*='C').

'S' bedeutet: Maschinen-spezifische Codierung (Standardcodierung, EBCDIC auf BS2000-Systemen, ASCII auf Unix-, Linux- und Windows-Systemen).

Ist *presentation_selector_type*='X' oder 'N', dann liefert UTM im Feld *presentation_selector_code* ein Leerzeichen zurück.

session_selector_type

gibt an, ob die Adresse der Partner-Anwendung einen Session-Selektor enthält und wie die Rückgabe in *session_selector* zu interpretieren ist.

'N' Steht für *NONE. Die Adresse der Partner-Anwendung enthält keinen Session-Selektor. Das Feld *session_selector* enthält Leerzeichen und *session_selector_lth*='0'.

'C' Die Angabe des Session-Selektors in *session_selector* ist als Character-String zu interpretieren. Es sind maximal die ersten 16 Byte von *session_selector* belegt.

'X' Der Session-Selektor in *session_selector* ist eine Hexadezimalzahl.

session_selector_lth

enthält die Länge des Session-Selektors *session_selector* in Byte. Ist *session_selector_lth*='0', dann hat die Adresse keine Session-Komponente. Sonst liegt der Wert von *session_selector_lth* zwischen '1' und '16'.

session_selector_code

gibt an, wie der Session-Selektor in *session_selector* codiert ist.

UTM liefert 'S' zurück, wenn der Session-Selektor als Character-String zurückgeliefert wird (*session_selector_type*='C').

'S' bedeutet: Maschinen-spezifische Codierung (Standardcodierung, EBCDIC auf BS2000-Systemen, ASCII auf Unix-, Linux- und Windows-Systemen).

Ist *session_selector_type*='X' oder 'N', dann liefert UTM im Feld *session_selector_code* ein Leerzeichen zurück.

transport_selector

enthält den Transport-Selektor der Partner-Anwendung. Der Transport-Selektor ist eine Komponente der Partneradresse. *transport_selector* enthält immer einen gültigen Wert, da jedem Kommunikationspartner ein Transport-Selektor zugeordnet sein muss. Der Transport-Selektor ist immer als Character-String zu interpretieren, er besteht aus 1 bis 8 abdruckbaren Zeichen.

network_selector

Netzkomponente (Network-Selektor) der Partneradresse.

BS2000-Systeme:

network_selector enthält den Namen des Rechners, auf dem sich die Partner-Anwendung befindet. Das ist der Name, mit dem der Rechner bei BCAM bekannt ist.

Unix-, Linux- und Windows-Systeme:

network_selector enthält den Namen des Partner-Rechners, mit dem UTM die IP-Adresse des Partner-Rechners im Name Service sucht.

Wenn der reale Rechnername länger als 8 Zeichen ist, dann gilt Folgendes:

- Das Feld *network_selector* enthält einen symbolischen lokalen Namen, der vom Transportsystem für diesen Rechner vergeben wurde.
- Der vollständige, bis zu 64 Zeichen lange Name kann dem Feld *network_selector_long* entnommen werden.

active gibt an, ob diese Verbindung aktiv gesetzt ist oder ob es sich um eine Ersatzverbindung

handelt, die derzeit inaktiv ist. Es ist möglich, mehrere Verbindungen zu einer Partner-Anwendung zu generieren. Zu einer Zeit kann jedoch nur eine dieser Verbindungen aktiv sein.

'Y' Die Verbindung ist aktiv gesetzt.

'N' Die Verbindung ist inaktiv.

map (nur auf Unix-, Linux- und Windows-Systemen)

gibt an, ob UTM für Benutzer-Nachrichten ohne Format-Kennzeichen (abstrakte Syntax UDT), die zwischen den Partner-Anwendungen ausgetauscht werden, eine Code-Konvertierung (ASCII <-> EBCDIC) durchführt.

'U' (USER)
UTM konvertiert die Benutzer-Nachrichten nicht, d.h. die Daten der Nachricht werden zwischen den die Partner-Anwendungen übertragen.

'1', '2', '3', '4' (SYS1 | SYS2 | SYS3 | SYS4)

UTM konvertiert die Benutzernachrichten gemäß den für die Code-Konvertierung bereitgestellten Konvertierungstabellen, siehe Abschnitt „Code-Konvertierung“ im openUTM-Handbuch „Anwendungen generieren“, d.h.:

- Vor dem Senden wird von ASCII nach EBCDIC konvertiert.
- Nach dem Empfangen wird von EBCDIC nach ASCII konvertiert.

Dabei geht UTM davon aus, dass die Nachrichten nur abdruckbare Zeichen enthalten.

Weitere Informationen zur Code-Konvertierung finden Sie im openUTM-Handbuch „Anwendungen programmieren mit KDCS“, Abschnitt "Code-Konvertierung".

In den folgenden Feldern liefert UTM die Komponenten der Transportadresse der Partner-Anwendung zurück. Siehe dazu auch openUTM-Handbuch „Anwendungen generieren“.

listener_port

enthält die Portnummer der Transportadresse der Partner-Anwendung.

listener_port='0' bedeutet, dass bei der KDCDEF-Generierung keine Portnummer angegeben wurde.

t_prot (nur auf Unix-, Linux- und Windows-Systemen)

enthält das Adressformat der Transportadresse. Das Adressformat gibt an, über welches Transportprotokoll die Kommunikation mit der Partner-Anwendung erfolgt.

'R' RFC1006, ISO-Transportprotokoll Klasse 0 über TCP/IP und Konvergenzprotokoll RFC1006.

Enthält *t_prot* ein Leerzeichen, dann wurde bei der KDCDEF-Generierung kein Adressformat definiert.

tsel_format (nur auf Unix-, Linux- und Windows-Systemen)

gibt das Format des T-Selektors der Partneradresse an:

'T' TRANSDATA-Format

'E' EBCDIC-Zeichenformat

'A' ASCII-Zeichenformat

Enthält *tsel_format* ein Leerzeichen, dann wurde bei der KDCDEF-Generierung kein Formatindikator definiert.

Zur Bedeutung der Adressformate siehe „Dokumentation zu PCMX“ im Abschnitt "openUTM-Dokumentation".

ip_addr (nur auf Unix-, Linux- und Windows-Systemen)

liefert die von UTM für diese Verbindung verwendete IP-Adresse aus der Objekttabelle der Anwendung, wenn es sich um eine IPv4-Adresse handelt.

Die Adresse benutzt UTM, um Verbindungen zur Partner-Anwendung aufzubauen. Die IP-Adresse wird von UTM mit Hilfe des generierten Rechnernamens (*network_selector*) beim Anwendungsstart aus dem Name-Service gelesen.

Eine IPv6-Adresse wird im Feld *ip_addr_v6* zurückgeliefert.

Existiert in den Objekttabellen für den Client keine IPv4-Adresse, liefert UTM in *ip_addr* Leerzeichen zurück.

ip_addr_v6 (nur auf Unix-, Linux- und Windows-Systemen)

liefert die von UTM für diese Verbindung verwendete IP-Adresse aus der Objekttabelle der Anwendung, wenn es sich um eine IPv6-Adresse handelt oder um eine IPv4-Adresse, die in ein IPv6-Format eingebettet ist.

Eine IPv4-Adresse wird im Feld *ip_addr* zurückgeliefert (siehe oben).

Existiert in den Objekttabellen für den Client keine IPv6-Adresse, liefert UTM in *ip_addr_v6* Leerzeichen zurück.

ip_v (nur auf Unix-, Linux- und Windows-Systemen)

gibt an, ob es sich bei der von openUTM für diese Verbindung verwendeten IP-Adresse um eine IPv4- oder um eine IPv6-Adresse handelt:

'V4' IPv4-Adresse.

'V6' IPv6-Adresse oder IPv4-Adresse, die in ein IPv6-Format eingebettet ist.

network_selector_long

Netzkomponente (Network-Selektor) der Partneradresse.

BS2000-Systeme:

network_selector_long enthält den Namen des Rechners, auf dem sich die Partner-Anwendung befindet. Das ist der Name, mit dem der Rechner bei BCAM bekannt ist.

Unix-, Linux- und Windows-Systeme:

network_selector_long enthält den Namen des Partsner-Rechners, mit dem UTM die IP-Adresse des Partner-Rechners im Name Service sucht.

11.3.1.22 kc_osi_lpap_str - Eigenschaften von OSI TP-Partner-Anwendungen

Für den Objekttyp KC_OSI_LPAP ist die Datenstruktur *kc_osi_lpap_str* definiert. In *kc_osi_lpap_str* liefert UTM bei KC_GET_OBJECT Folgendes zurück:

- die logischen Eigenschaften einer OSI TP-Partner-Anwendung.
  Die logischen Eigenschaften einer OSI TP-Partner-Anwendung werden bei der KDCDEF-Generierung definiert, indem ein OSI-LPAP-Partner erzeugt und dieser Partner-Anwendung zugeordnet wird.
- den Zustand der Verbindung zu der Partner-Anwendung.
- Statistikinformationen über die Auslastung der Verbindung.

mod[1]	Datenstruktur kc_osi_lpap_str
-	char ol_name[8];
-	char application_context[8];
-	char application_entity_qualifier[8];
-	char application_process_title[10][8];
-	char association_names[8];
-	char associations[5];
x(GPD)	char auto_connect_number[5];
-	char contwin[5];
-	char kset[8];
x(GPD)	char idletime_sec[5];
x(GPD)	char state;
-	char permit;
-	char qlev[5];
-	char termn[2];
x(A)	char connect_number[5];
x(IR)	char quiet_connect;
-	char osi_con[8];
-	char out_queue[5];
-	char ass_kset[8];
-	char nbr_dputs[10];
-	char master[8];
-	char bundle;
-	char out_queue_ex[10];
x(GPD)	char dead_letter_q;

[1] Feldinhalt mit KC_MODIFY_OBJECT modifizierbar; siehe Abschnitt "obj_type=KC_OSI_LPAP"

Die Felder der Datenstruktur haben die folgende Bedeutung:

ol_name

enthält den Namen des OSI-LPAP-Partners der Partner-Anwendung. Über diesen Namen wird die Partner-Anwendung von den Teilprogrammen der lokalen Anwendung angesprochen. Der Name besteht aus maximal 8 alphanumerischen Zeichen.

application_context

gibt an, welcher Application Context von der Partner-Anwendung verwendet wird. Näheres zu Application Context siehe openUTM-Handbuch „Anwendungen generieren“, KDCDEF-Anweisung APPLICATION-CONTEXT.

application_entity_qualifier

enthält den Application Entity Qualifier (AEQ) der Partner-Anwendung. Der Application-Entity-Qualifier wird zusammen mit dem Application Process Title zur Adressierung einer Partner-Anwendung bei heterogener Kopplung verwendet.
Der Application-Entity-Qualifier ist eine positive ganze Zahl zwischen 1 und 67108863 (= 2^{26}-1).

Näheres zum Application Entity Qualifier finden Sie im openUTM-Handbuch „Anwendungen generieren“.

application_entity_qualifier='0' bedeutet, dass für die Partner-Anwendung kein AEQ definiert ist.

application_process_title

enthält den Application Process Title (APT) der Partner-Anwendung. Der APT wird zusammen mit dem Application-Entity-Qualifier zur Adressierung einer Partner-Anwendung bei heterogener Kopplung verwendet.

Ein APT besteht aus mindestens 2, maximal aber aus 10 Komponenten. Die einzelnen Komponenten sind positive ganze Zahlen.

Bei einem Application Process Title ist sowohl die Position der einzelnen Komponenten als auch ihre Anzahl relevant, z.B. bezeichnen (1,2,3), (1,2,3,0,0) und
(0,1,2,3,0) verschiedene Application Process Titles. Näheres zum Application Process Title finden Sie im openUTM-Handbuch „Anwendungen generieren“.

UTM liefert pro Komponente des APT ein Feldelement zurück, d.h. die Anzahl der belegten Feldelemente in *application_process_title* entspricht der generierten Anzahl der Komponenten. Die restlichen Feldelemente sind mit binär null versorgt.

Wurde für die Partner-Anwendung kein APT generiert, dann sind alle Feldelemente von *application_process_title* mit binär null versorgt.

association_names

enthält das Präfix der Namen, die den logischen Verbindungen (Associations) zur Partner-Anwendung innerhalb der lokalen Anwendung zugeordnet werden.

Der Name der Verbindungen setzt sich zusammen aus dem Wert von *association_names* als Präfix und einer Laufnummer. Die Laufnummer liegt zwischen 1 und der generierten Anzahl der parallelen Verbindungen (Feld *associations*). Der gesamte Name für eine Verbindung kann bis zu 8 Zeichen lang sein. Die maximale Länge von *association_names* ist deshalb abhängig von der Anzahl paralleler Verbindungen in *associations*.

Beispiel:
association_names='ASSOC' und *associations*='10'; dann haben die Verbindungen zur Partner-Anwendung, die dem OSI-LPAP-Partner zugeordnet sind, folgende Namen: ASSOC01, ASSOC02, ..., ASSOC10.

associations

enthält die maximale Anzahl der parallelen Verbindungen zur Partner-Anwendung. Die maximal mögliche Anzahl paralleler Verbindungen zu einer Partner-Anwendung ist abhängig vom verwendeten Transportsystem und von der Größe des Namensraums der UTM-Anwendung (siehe openUTM-Handbuch „Anwendungen generieren“).

auto_connect_number

enthält die Anzahl der Verbindungen, die beim Start der lokalen Anwendung zur Partner-Anwendung automatisch aufgebaut werden, sofern die Partner-Anwendung zu diesem Zeitpunkt verfügbar ist. Der Aufbau einer Verbindung beim Start der Anwendung kann sowohl in der lokalen Anwendung als auch bei der Partner-Anwendung angefordert werden. Dadurch erreicht man, dass die Verbindung automatisch aufgebaut wird, sobald beide Partner verfügbar sind.

auto_connect_number='0' bedeutet, dass kein automatischer Verbindungsaufbau stattfindet.

Minimalwert: '0'
Maximalwert: maximale Anzahl der parallelen Verbindungen (*associations*)

contwin

enthält die Anzahl der Verbindungen, für die die lokale Anwendung der Contention Winner ist. Für die restlichen Verbindungen (Differenz: *associations* - *contwin*) ist die lokale Anwendung Contention Loser.

Der Contention Winner einer Verbindung übernimmt die Verwaltung der Verbindung. Aufträge können aber sowohl vom Contention Winner als auch vom Contention Loser gestartet werden. Im Konfliktfall, wenn beide Kommunikationspartner gleichzeitig einen Auftrag starten wollen, wird die Verbindung vom Auftrag des Contention Winner belegt.

kset

enthält den Namen des Keysets, das die maximalen Zugriffsrechte der OSI TP-Partner-Anwendung in der lokalen Anwendung angibt.

Übergibt der OSI TP-Partner beim Aufbau einer Association eine Benutzerkennung,dann wird das Keyset in *kset* wirksam. Die Zugriffsrechte für diese Association entsprechen den Keycodes, die sowohl in *kset* als auch in dem Keyset der Benutzerkennung enthalten sind.

Übergibt die OSI TP-Partner-Anwendung für die Association keine Benutzerkennung an openUTM, dann ergeben sich die Zugriffsrechte für diese Association aus der Schnittmenge der Keycodes in *kset* und *ass_kset* (minimale Zugriffsrechte).

Ist der Partner-Anwendung kein Keyset zugeordnet, dann ist *kset* mit Leerzeichen belegt.

idletime_sec

enthält die Zeit in Sekunden, die sich eine Association zur Partner-Anwendung maximal im Leerlauf-Zustand (Idle-Zustand) befinden darf, bevor UTM die Verbindung zur Partner-Anwendung abbaut. Leerlauf-Zustand heißt: die Verbindung ist nicht durch einen Auftrag belegt.

idletime_sec='0' bedeutet, dass der Leerlauf-Zustand nicht überwacht wird. Die Verbindung bleibt bestehen, bis der Verbindungsabbau explizit angefordert wird.

Minimalwert: '60'
Maximalwert: '32767'

state

gibt an, ob der OSI-LPAP-Partner zur Zeit gesperrt ist.

'Y' Der OSI-LPAP-Partner ist nicht gesperrt. Es können Verbindungen zwischen der Partner-Anwendung und der lokalen Anwendung aufgebaut werden oder es bestehen bereits Verbindungen.

'N' Der OSI-LPAP-Partner ist gesperrt. Es können keine Verbindungen zwischen Partner-Anwendung und lokaler Anwendung aufgebaut werden.

permit gibt an, welche Berechtigungsstufe die Partner-Anwendung innerhalb der lokalen Anwendung hat.

'A' (ADMIN)
Die Partner-Anwendung hat die Administrationsberechtigung. Sie darf alle Administrationsfunktionen in der lokalen Anwendung ausführen.

'N' (NONE)
Die Partner-Anwendung hat keine Administrationsberechtigung.

Nur auf BS2000-Systemen:

- Ist die lokale Anwendung eine UTM-Anwendung auf einem BS2000-System, dann darf die Partner-Anwendung auch keine UTM-SAT-Administrationsfunktionen ausführen.

'B' (BOTH, nur auf BS2000-Systemen)
Die Partner-Anwendung darf Administrations- und UTM-SAT-Administrationsfunktionen in der lokalen Anwendung ausführen.

'S' (SAT, nur auf BS2000-Systemen)

Die Partner-Anwendung hat nur UTM-SAT-Administrationsberechtigung.
Sie darf Preselection-Funktionen in der lokalen Anwendung ausführen, d.h. sie kann die SAT-Protokollierung bestimmter Ereignisse ein- bzw. ausschalten.

qlev (**q**ueue **lev**el)

qlev gibt an, wieviele Asynchron-Nachrichten maximal in der Message Queue des OSI-LPAP-Partners stehen dürfen. Wird dieser Schwellwert überschritten, so werden weitere Asynchron-Aufträge an diesen OSI-LPAP-Partner abgewiesen (d.h. bei folgenden APRO-AM-Aufrufen wird '40Z' zurückgeliefert).

termn

enthält das Kennzeichen für die Art des Kommunikationspartners. Das Kennzeichen wird im KB-Kopf der Auftragnehmer-Vorgänge eingetragen, die von der Partner-Anwendung in der lokalen Anwendung gestartet wurden. Das Kennzeichen wird vom Anwender definiert und dient dazu, die Kommunikationspartner in Gruppen bestimmten Typs einzuteilen. Es wird von UTM nicht ausgewertet. Das Kennzeichen ist maximal 2 Zeichen lang.

connect_number

enthält die Anzahl der parallelen Verbindungen, die derzeit zur Partner-Anwendung aufgebaut sind bzw. aufgebaut werden sollen.

connect_number='0' bedeutet, dass zur Zeit keine Verbindung zur Partner-Anwendung existiert bzw. alle bestehenden Verbindungen abgebaut werden sollen.

Minimalwert: '0'
Maximalwert: die Anzahl in *associations*

quiet_connect

gibt an, ob für die Partner-Anwendung die Eigenschaft „QUIET“ gesetzt ist bzw. gesetzt wird. QUIET bedeutet, dass UTM alle Verbindungen zur Partner-Anwendung abbaut, sobald diese nicht mehr durch Dialog- oder Asynchron-Aufträge belegt sind. Für die Partner-Anwendung werden keine neuen Dialog-Aufträge mehr angenommen.

'Y' Für die Partner-Anwendung ist die Eigenschaft QUIET gesetzt.

'N' Die Eigenschaft QUIET ist nicht gesetzt.

osi_con

enthält den Namen der Transportverbindung, über die mit der Partner-Anwendung kommuniziert wird, d. h. über diese Transportverbindung werden alle Verbindungen (Associations) mit der Partner-Anwendung abgewickelt. Der Name wird der Transportverbindung bei der KDCDEF-Generierung zugeordnet (OSI-CON-Anweisung, die dem OSI-LPAP-Partner zugeordnet ist). *osi_con* bezeichnet die Transportverbindung, die zur Zeit aktiv gesetzt ist, d.h. nicht als Ersatzverbindung deaktiviert ist (siehe openUTM-Handbuch „Anwendungen generieren“).

out_queue

Anzahl der Nachrichten in der Message Queue des OSI-LPAP-Partners, die noch an die Partner-Anwendung gesendet werden müssen.

Ist die Anzahl der Nachrichten größer als 99999, wird die Zahl nur unvollständig dargestellt. Deshalb sollte das Feld *out_queue_ex* verwendet werden, weil hier auch größere Zahlen vollständig abgebildet werden.

ass_kset

nur relevant, wenn die Anwendung mit Benutzerkennungen generiert ist. *ass_kset* enthält den Namen des Keysets, das die minimalen Zugriffsrechte des OSI-TP-Partners angibt, die die Partner-Anwendung in der lokalen Anwendung ausüben kann.

Das in *ass_kset* angegebene Keyset wird dann wirksam, wenn die Partner-Anwendung beim Aufbau der Association keine Benutzerkennung an UTM übergibt (siehe auch *kset*). *ass_kset* beschreibt die Zugriffsrechte des Association-Users.

Standard: kein Keyset, d.h. es gelten die Zugriffsrechte in *kset*.

nbr_dputs

Anzahl anstehender zeitgesteuerter Aufträge für dieses OSI-LPAP, deren Startzeitpunkt noch nicht erreicht ist.

master

Zugehöriges Master-OSI-LPAP, falls es sich bei dem OSI-LPAP um ein Slave-OSI-LPAP in einem OSI-LPAP-Bündel handelt.

bundle

Angabe, ob das OSI-LPAP zu einem OSI-LPAP-Bündel gehört.

'N' Das OSI-LPAP gehört nicht zu einem OSI-LPAP-Bündel.

'M' Das OSI-LPAP ist der Master-OSI-LPAP in einem OSI-LPAP-Bündel. In diesem Fall gilt:

- bei KC_MODIFY_OBJECT kann nur das Feld *state* geändert werden.
- bei KC_GET_OBJECT sind nur die Felder *ol_name*, *application_context*, *state* und *bundle* relevant.

'S' Das OSI-LPAP ist ein Slave-OSI-LPAP in einem OSI-LPAP-Bündel.

out_queue_ex

siehe out_queue.

dead_letter_q

gibt an, ob eine Asynchron-Nachricht an den OSI-LPAP-Partner in die Dead Letter Queue gesichert werden soll, wenn sie wegen eines permanenten Fehlers nicht gesendet werden konnte.

'Y' Asynchron-Nachrichten an diesen OSI-LPAP-Partner, die wegen eines permanenten Fehlers nicht gesendet werden konnten, werden in die Dead Letter Queue gesichert, sofern (bei Message-Komplexen) kein negativer Quittungsauftrag definiert wurde.

'N' Asynchron-Nachrichten an diesen OSI-LPAP-Partner, die wegen eines permanenten Fehlers nicht gesendet werden konnten, werden gelöscht.

11.3.1.23 kc_program_str - Teilprogramme und VORGANG-Exits

Für den Objekttyp KC_PROGRAM ist die Datenstruktur *kc_program_str* definiert. Bei KC_GET_OBJECT liefert UTM in *kc_program_str* Informationen über die Teilprogramme und VORGANG-Exits der UTM-Anwendung zurück.

Teilprogramme können dynamisch mit KC_CREATE_OBJECT erzeugt und mit KC_DELETE_OBJECT gelöscht werden.

Datenstruktur kc_program_str
`char pr_name[32];`
`char compiler;`
`char load_module[32];`
`char load_mode;`
`char poolname[50];` (nur auf BS2000-Systemen)
`char lib[54];`
`char deleted;`

Die Felder der Datenstruktur haben die folgende Bedeutung:

pr_name

enthält den Namen des Teilprogramms.

In UTM-Anwendungen auf BS2000-Systemen liefert UTM den ENTRY- bzw. CSECT-Namen des Teilprogramms zurück.

compiler

gibt an, mit welches Laufzeitsystem bzw. welcher Compiler dem Teilprogramm in der Generierung zugeordnet wurde. Die Werte, die UTM zurückliefert, sind abhängig von der Betriebssystem-Plattform, auf der das Teilprogramm abläuft.

In UTM-Anwendungen auf BS2000-Systemen wird für alle Teilprogramme, die ILCS unterstützen, *compiler*='I' zurückgeliefert.

In UTM-Anwendungen auf BS2000-Systemen sind folgende Werte möglich:

- 'I' für ILCS (Inter Language Communication Services)
- 'A' für den Assembler-Compiler ASSEMB
- 'C' für den C-Compiler (nur beim KC_CREATE_OBJECT-Aufruf)
- '1' für den COBOL-Compiler COB1
- 'F' für den Fortran-Compiler FOR1
- 'X' für PASCAL-XT

'P' für PLI1

'S' für SPL4

In einer UTM-Anwendung auf Unix-, Linux- und Windows-Systemen sind folgende Werte möglich:

'C' für den C-Compiler

'2' für den COBOL-Compiler von Micro Focus

'3' für den NetCOBOL-Compiler von Fujitsu

'+' für den C++-Compiler

load_module

enthält den Namen des Lademoduls (BS2000-Systeme) bzw. des Shared Objects/DLLs (Unix-, Linux- und Windows-Systeme), in dem das Teilprogramm eingebunden ist.

Ist das Teilprogramm keinem Lademodul bzw. Shared Object/DLL zugeordnet, dann liefert UTM Leerzeichen zurück.

load_mode

enthält den Lademodus des Teilprogramms bzw. des Lademoduls/Shared Objects/DLL, in dem das Teilprogramm gebunden ist. Der Lademodus gibt an, wann und wohin das Teilprogramm bzw. das Lademodul/Shared Object/DLL geladen wird.

'U' (START**U**P)
Das Teilprogramm bzw. Lademodul/Shared Object/DLL wird beim Start der Anwendung als eigenständige Einheit nachgeladen.

'O' (**O**NCALL)
Das Lademodul bzw. Shared Object/DLL wird als eigenständige Einheit nachgeladen, wenn eines seiner Teilprogramme oder VORGANG-Exits erstmalig aufgerufen wird.

Bei Lademodulen einer UTM-Anwendung unter einem BS2000-System können in *load_mode* zusätzlich folgende Werte zurückgeliefert werden:

'S' (**S**TATIC)
Das Teilprogramm bzw. das Lademodul ist statisch in das Anwendungsprogramm eingebunden.

'P'

(**P**OOL)
Das Teilprogramm bzw. das Lademodul wird beim Start der Anwendung in einen Common Memory Pool (siehe *poolname*) geladen. Das Lademodul besteht nur aus einer public Slice (keine private Slice).

'T' (POOL/S**T**ARTUP)
Die public Slice des Lademoduls wird beim Start der Anwendung in einen Common Memory Pool (siehe *poolname*) geladen. Die zu dem Lademodul gehörige private Slice wird anschließend in den prozesslokalen Speicher geladen (private Slice mit Lademodus STARTUP).

'C' (POOL/ON**C**ALL)
Die public Slice des Lademoduls wird beim Start der Anwendung in einen Common Memory Pool (siehe *poolname*) geladen. Die zu dem Lademodul gehörige private Slice wird in den prozesslokalen Speicher geladen, sobald ein Teilprogramm aufgerufen wird, das diesem Lademodul zugeordnet ist (private Slice mit Lademodus ONCALL).

poolname (nur auf BS2000-Systemen)

Bei *load_mode*='P', 'T' oder 'C' enthält l*poolname* den Namen des Common Memory Pools zurück, in den das Teilprogramm bzw. die public Slice seines Lademoduls beim Start der Anwendung geladen wird.

Bei *load_mode* ! = 'P', 'T' oder 'C' ist *poolname* mit Leerzeichen versorgt.

lib

lib enthält Folgendes:

- in UTM-Anwendung unter einem BS2000-System, die ohne Lademodule generiert ist, die Objektmodulbibliothek, aus der das Teilprogramm nachgeladen oder gebunden wird.
- in einer UTM-Anwendung unter einem BS2000-System, die mit Lademodulen generiert ist, die Programm-Bibliothek, aus der das Lademodul nachgeladen wird.
- in einer UTM-Anwendung auf Unix-, Linux- und Windows-Systemen, die mit Shared Objects generiert ist, das Dateiverzeichnis (directory), in dem das Shared Object/DLL abgelegt ist.

deleted

gibt bei KC_GET_OBJECT an, ob das Teilprogramm durch die dynamische Administration aus der Konfiguration gelöscht wurde oder nicht.

'Y' Das Teilprogramm wurde gelöscht. Der Name ist gesperrt. Es kann kein neues Teilprogramm mit diesem Namen eingetragen werden.

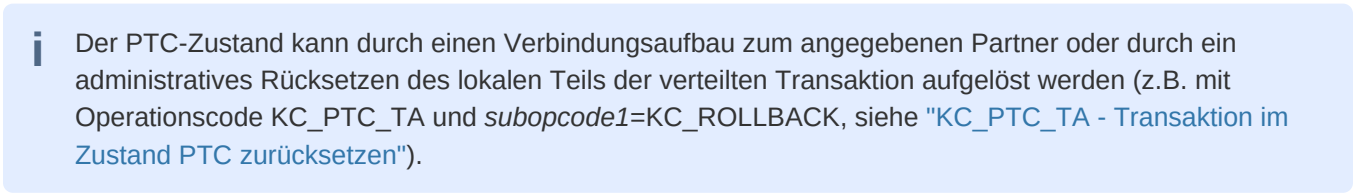
'N' Das Teilprogramm wurde nicht aus der Konfiguration gelöscht.

11.3.1.24 kc_ptc_str - Transaktionen im Zustand PTC

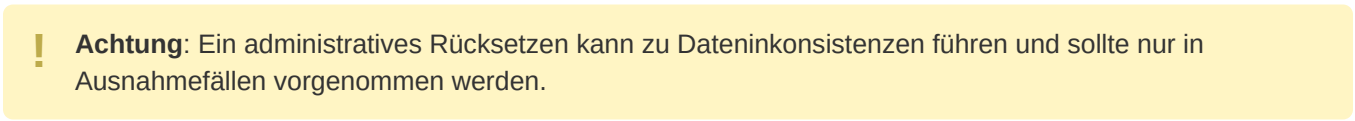
Für den Objekttyp KC_PTC ist die Datenstruktur *kc_ptc_str* definiert. In *kc_ptc_str* werden bei KC_GET_OBJECT alle verteilten Transaktionen angezeigt, die sich im Zustand PTC (prepare to commit) befinden und bei denen keine Verbindung (LU6.1-Session oder OSI-TP) zum Commit Coordinator besteht. Der Commit Coordinator ist die Partner-Anwendung, die über das Ergebnis der Transaktion entscheidet.
Mit *prepare to commit* bezeichnet man den Zustand einer Transaktion, in dem ein Partner bereits Transaktionsende eingeleitet hat und auf die Entscheidung eines Kommunikationspartners zum Transaktionsausgang wartet. In diesem Zustand werden von der lokalen Transaktion Sperren auf Ressourcen der Anwendung oder einer Datenbank gehalten.

Dabei liefert openUTM Folgendes zurück:

- Informationen über die Transaktion
- Auftraggeber-Benutzerkennung der verteilten Transaktion
- Name des Partners (LPAP- bzw. OSI-LPAP-Partner)
- Name der Session (bei LU6.1)



Der PTC-Zustand kann durch einen Verbindungsaufbau zum angegebenen Partner oder durch ein administratives Rücksetzen des lokalen Teils der verteilten Transaktion aufgelöst werden (z.B. mit Operationscode KC_PTC_TA und *subopcode1*=KC_ROLLBACK, siehe "KC_PTC_TA - Transaktion im Zustand PTC zurücksetzen").



Achtung: Ein administratives Rücksetzen kann zu Dateninkonsistenzen führen und sollte nur in Ausnahmefällen vorgenommen werden.

Datenstruktur kc_ptc_str
`struct kc_ptc_id_str ptc_ident;`
`char ptc_user[8];`
`char ptc_lpap[8];`
`char ptc_lses[8];`
`char ptc_user_type;`

Die Felder der Datenstruktur haben die folgende Bedeutung:

ptc_ident

Gibt die Informationen zur Transaktion in dem Element *ptc_ident* vom Typ *kc_ptc_id_str* zurück:

struct kc_ptc_id_str
`char vg_indx[10];`
`char vg_nr[10];`
`char ta_nr_in_vg[5];`

vg_indx ist der Index des Vorgangs, *vg_nr* die Nummer des Vorgangs und *ta_nr_in_vg* die Nummer der Transaktion im Vorgang.

Die Struktur muss beim Operationscode KC_PTC_TA mit *subopcode1*=KC_ROLLBACK im Identifikationsbereich übergeben werden, wenn man die Transaktion zurücksetzen will.

ptc_user

gibt die Auftraggeber-Benutzerkennung der verteilten Transaktion an.

Bei OSI TP enthält das Feld den Namen einer OSI TP Association.

Bei LU6.1 kann das Feld den Namen eines Benutzers (USER), einer LU6.1-Session (LSES) oder 8 Leerzeichen enthalten:

- Enthält das Feld 8 Leerzeichen, so ist die Transaktion in einem asynchronen LU6.1-Auftraggeber-Vorgang im Zustand PTC.
- Enthält das Feld den Namen eines Benutzers, so ist die Transaktion im obersten LU6.1-Auftraggeber-Dialogvorgang im Zustand PTC.
- Ist der Inhalt des Felds ungleich dem Inhalt des Feldes *ptc_lses*, so ist die Transaktion in einem LU6.1-Auftraggeber-Vorgang im Zustand PTC.
- Ist der Inhalt des Felds gleich dem Inhalt des Feldes *ptc_lses*, so ist die Transaktion in einem LU6.1-Auftragnehmer-Vorgang im Zustand PTC.

ptc_lpap

gibt den LPAP- bzw. OSI-LPAP-Namen des Partners an, der über den Ausgang der Transaktion entscheidet (Commit Coordinator).

ptc_lses

gibt bei LU6.1-Verbindungen den Session-Namen des Partners an, der über den Ausgang der Transaktion entscheidet (Commit Coordinator).

Bei einer PTC-Transaktion im Auftragnehmer hat *ptc_lses* denselben Inhalt wie *ptc_user*.

Bei OSI-TP-Verbindungen enthält dieses Feld Leerzeichen.

ptc_user_type

Typ des Eintrags in Feld *ptc_user*:

U	User
L	LU6.1-Session
O	OSI TP Association
Leerzeichen	wenn das Feld *ptc_user* leer ist

11.3.1.25 kc_pterm_str - Clients und Drucker

Für den Objekttyp KC_PTERM ist die Datenstruktur *kc_pterm_str* definiert.
Bei KC_GET_OBJECT liefert UTM in *kc_pterm_str* die folgenden Informationen zurück:

- Eigenschaften von Clients und Druckern, die statisch oder dynamisch in die Konfiguration der Anwendung eingetragen wurden.
- Eigenschaften von Clients, die zur Zeit über einen LTERM-Pool oder Multiplexanschluss mit der Anwendung verbunden sind.
- Eigenschaften und Zustand der Verbindung zu dem entsprechenden Client oder Drucker.
- Zu den einzelnen Clients/Druckern Statistikinformationen über Auslastung der Verbindung und Anforderungen an die Anwendung.

Clients und Drucker können dynamisch mit KC_CREATE_OBJECT erzeugt, mit KC_DELETE_OBJECT gelöscht oder mit KC_MODIFY_OBJECT modifiziert werden.

mod[1]	**Datenstruktur kc_pterm_str**	**siehe**[2]
-	`char pt_name [8];`	pt_name
-	`char pronam [8];`	pronam
-	`char bcamappl[8];`	bcamappl
-	`char ptype[8];`	ptype
-	`char ptype_fotyp[8];`	PRINTER
-	`char ptype_class[40];`	PRINTER
x(PD)	`char lterm[8];`	lterm
x(GPD)	`char auto_connect;`	auto_connect
x(GPD)	`char state;`	state
-	`char cid[8];`	cid
-	`char map;`	map
-	`char termn[2];`	termn
-	`char protocol;` (nur auf BS2000-Systemen)	protocol
-	`char usage_type;` (nur auf BS2000-Systemen)	usage_type
-	`char listener_port[5];`	listener_port
-	`char t_prot;`	t_prot
-	`char tsel_format;` (nur auf Unix-, Linux- und Windows-Systemen)	tsel_format

mod[1]	Datenstruktur kc_pterm_str	siehe[2]
x(A)	`char connect_mode;`	connect_mode
-	`char pool;`	pool
-	`char mux;` (nur auf BS2000-Systemen)	mux
-	`char contime_min[10];`	contime_min
-	`char letters[10];`	letters
-	`char conbad[5];`	conbad
-	`char deleted;`	deleted
X(GPD)	`char idletime[5];`	idletime
-	`char encryption_level;`	encryption_level
-	`char ip_addr[15];`	ip_addr
-	`char curr_encryption;`	curr_encryption
-	`char t_mode;` [3]	
-	`char usp_hdr;`	usp_hdr
-	`char ip_addr_v6[39];`	ip_addr_v6
-	`char ip_v[2];`	ip_v
-	`char pronam_long[64];`	pronam_long

1 Feldinhalt mit KC_MODIFY_OBJECT modifizierbar; siehe "obj_type=KC_PTERM".

2 Die Bedeutung der Felder ist auf der in dieser Spalte angegebenen Stelle beschrieben.

3 UTM-internes Feld; der Feldinhalt ist irrelevant und wird im Folgenden nicht beschrieben.

Die Felder der Datenstruktur haben die folgende Bedeutung:

pt_name

enthält den Namen des Client oder Druckers. Der Client/Drucker ist dem Transportsystem (BCAM, PCMX) unter diesem (symbolischen) Namen bekannt.

pronam

Name des Rechners, auf/an dem sich der Client bzw. Drucker befindet.

In UTM-Anwendungen auf BS2000-Systemen ist *pronam* immer belegt. Bei einem RSO-Drucker enthält *pronam* den Wert '*RSO'.

In UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen enthält *pronam* bei einem lokalen Client oder einem Drucker Leerzeichen.

Wenn der reale Rechnername länger als 8 Zeichen ist, dann gilt Folgendes:

- Das Feld *pronam* enthält einen symbolischen lokalen Namen, der vom Transportsystem für diesen Rechner vergeben wurde.
- Der vollständige, bis zu 64 Zeichen lange Name kann dem Feld *pronam_long* entnommen werden.
- War noch keine Verbindung aufgebaut, dann enthält *pronam* Leerzeichen.

bcamappl

Name der lokalen UTM-Anwendung, über den die Verbindung zu dem Client/Drucker aufgebaut wird.

Bei Terminals und Druckern enthält *bcamappl* immer den Namen der Anwendung, der bei der KDCDEF-Generierung in MAX APPLINAME festgelegt wurde.

Bei UPIC-Clients und TS-Anwendungen enthält *bcamappl* immer (auch wenn z.Zt. keine Verbindung besteht) den Anwendungsnamen, der dem Client beim Eintragen in die Konfiguration zugeordnet wurde.

Nur auf BS2000-Systemen:

- Bei Clients, die über einen Multiplexanschluss mit der Anwendung verbunden sind, enthält *bcamappl* für die Dauer der Verbindung den Anwendungsnamen, der dem Multiplexanschluss beim Eintragen in die Konfiguration zugeordnet wurde.

ptype

Typ des Clients oder Druckers.

BS2000-Systeme

Für Clients/Drucker, die an eine UTM-Anwendung unter einem BS2000-System angeschlossen sind, wird entweder explizit der Partnertyp zurückgeliefert oder der Wert '*ANY' oder der Wert '*RSO'. Die unterstützten Partnertypen sind in der folgenden Tabelle aufgeführt:

ptype	Typ des Client/Druckers	Feld termn
*ANY	Der Client wurde ohne genaue Angabe des Gerätetyps in die Konfiguration aufgenommen. In diesem Fall nimmt UTM den Gerätetyp des Clients beim Verbindungsaufbau aus dem Benutzerdienstprotokoll. Erst dann wird entschieden, ob der Partnertyp unterstützt wird. Vorteil von *ptype*='*ANY': Sie können Clients in die Konfiguration aufnehmen, ohne ihren Typ zu kennen. Darüber hinaus wird die Pflege der Konfiguration erleichtert, denn auch wenn der Typ z.B. in der Terminalemulation geändert wird, kann dieser Client weiterhin eine Verbindung zu der Anwendung aufbauen, ohne dass Sie die Konfiguration der Anwendung ändern müssen.	Wurde die Terminalmnemonic bei der Konfiguration nicht explizit angegeben, dann wird beim Verbindungsaufbau die Standard-Terminalmnemonic des Partnertyps genommen. Sonst steht hier der bei der Konfiguration angegebene Wert.

ptype	Typ des Client/Druckers	Feld termn
T100	Fernschreiber T100	C0
T1000	Fernschreiber T1000	E1
T8103	8103	FD
T8110	8110	C2
T8121V12	Drucker 8121 an 8112	F7
T8122V12	Drucker 8122 an 8112	F8
T8124	Drucker 8124	FC
T8151	DSS 8151	F1
T8152	DSS 8152	F2
T8160	DSS 8160	F4
T8162	DSS 8162	F6
T8167	DSS 8167	FB
T9748 [1]	DSS 9748	FE
T9749	DSS 9749	FE
T9750 [1]	DSS 9750	FE
T9751	DSS 9751	FE
T9752	DSS 9752	FF
T9753	DSS 9753	FE
T9754	DSS 9754	FI
T9755 [2]	DSS 9755	FG
T9756[2]	DSS 9756	FG
T9763	DSS 9763	FH
T9770	DSS 9770	FK
T9770R	DSS 9770R	FK
T3270	DSS 3270 (IBM)	FL
THCTX28	DSS X28 (TELETYPE)	C5

ptype	Typ des Client/Druckers	Feld termn
TVDTX28	DSS X28 (VIDEO)	C6
TPT80	Datenstation PT80	C4
T9001	Drucker 9001	C7
T9002	Drucker 9002	FA
T9003	Drucker 9003	F9
T9004	Drucker 9004	FD
T9001-3	Drucker 9001-3	CA
T9001-89	Drucker 9001-893	CB
T9011-18	Drucker 9011-18	CC
T9011-19	Drucker 9011-19	CD
T9012	Drucker 9012	CE
T9013	Drucker 9013	C9
T9021	Drucker 9021	CH
T9022	Drucker 9022	CF
T3287	Drucker 3287	CG
*RSO	Drucker, der über RSO unterstützt wird. Anstatt eine Transportverbindung aufzubauen, reserviert UTM den Drucker bei RSO und übergibt die zu druckende Nachricht an RSO.	RS
THOST	Intelligente Datenstation	A3
APPLI	Transportsystem-Anwendung, die keine Socket-Anwendung ist, z.B.: DCAM-, CMX- oder UTM-Anwendung.	A1
UPIC-R	UPIC-Client	A5
SOCKET	USP Socket-Anwendung	A7
SOCKET	HTTP-Client	A8
SOCKET	Secure USP Socket-Anwendung	A9
SOCKET	HTTPS-Client	AA

[1] T9748 und T9750 bezeichnen den gleichen Terminaltyp.

[2] T9755 und T9756 bezeichnen den gleichen Terminaltyp.

In welcher VTSU-Version die einzelnen Terminals unterstützt werden, ist dem DCAM-, FHS- bzw. dem TIAM-Handbuch zu entnehmen.

Wird ein Terminal von VTSU nicht unterstützt, so weist UTM eine Verbindungsanforderung von diesem Terminal zurück. UTM erzeugt die Meldungen K064 und K107.

Unix, Linux- und Windows-Systeme

In einer UTM-Anwendung auf Unix-, Linux- und Windows-Systemen kann *ptype* folgende Werte enthalten:

ptype	Typ des Client/Druckers	Feld termn
TTY	Der Client ist ein Terminal. Standardwert	F1
PRINTER	*ptype*='PRINTER' ist nur auf Unix- und Linux-Systemen relevant und ist unter openUTM auf Windows-Systemen nicht erlaubt. Die Bedeutung von *ptype*='PRINTER' ist abhängig vom Inhalt des Feldes *ptype_fotyp*.	F2
	pt_name spezifiziert den Namen eines Druckers, auf dem mit dem Spool auf Unix- und Linux-Systemen ausgedruckt wird.	
	ptype_fotyp und *ptype_class* sind entweder mit Leerzeichen belegt oder enthalten den passenden Druckertyp bzw. die Druckergruppe von *pt_name*.	
APPLI	Der Client ist eine TS-Anwendung, die nicht die Socket-Schnittstelle benutzt (z.B. UTM-, CMX-, DCAM-Anwendung)	A1
UPIC-L	Der Client ist eine lokale UPIC-Client-Anwendung.	A5
UPIC-R	Der Client ist eine ferne (remote) UPIC-Client-Anwendung.	A5
SOCKET	USP Socket-Anwendung	A7
SOCKET	HTTP-Client	A8
SOCKET	Secure USP Socket-Anwendung	A9
SOCKET	HTTPS-Client	AA

lterm

Name des LTERM-Partners, der diesem Client/Drucker zugeordnet ist.

auto_connect

gibt an, ob die Verbindung zu dem Client/Drucker beim Start der Anwendung automatisch aufgebaut wird:

'Y' Beim Start der Anwendung versucht UTM, die Verbindung automatisch aufzubauen, wenn der Client/Drucker zum Zeitpunkt des Starts der lokalen Anwendung verfügbar ist.

'N' Kein automatischer Verbindungsaufbau beim Start.

state

gibt an, ob der Client oder Drucker zur Zeit gesperrt ist.

'Y' Der Client/Drucker ist nicht gesperrt, d.h. sofern der LTERM-Partner, der diesem Client/Drucker zugeordnet ist, nicht gesperrt ist, können Verbindungen zwischen dem Client/Drucker und der lokalen Anwendung aufgebaut werden oder es bestehen bereits Verbindungen.

'N' Der Client/Drucker ist gesperrt. Es können keine Verbindungen zwischen Client/Drucker und lokaler Anwendung aufgebaut werden.

cid (**c**ontrol **id**entification)

ist nur belegt, wenn Informationen über einen Drucker abgefragt werden.
cid enthält die Drucker-Id (CID), sofern dem Drucker beim Eintragen in die Konfiguration eine CID zugeordnet wurde.

Die CID hat folgende Funktion:

- Über die CID kann der Drucker an der Programmierschnittstelle zur Drucksteuerung identifiziert werden.
- Ist der Drucker einem Druckersteuer-LTERM zugeordnet, dann wird der Drucker bei der Administration vom Druckersteuer-LTERM aus über die CID identifiziert.

map

gibt an, ob UTM für Benutzer-Nachrichten, die zwischen den Partner-Anwendungen ausgetauscht werden, eine Code-Konvertierung (ASCII <-> EBCDIC) durchführt. Benutzer-Nachrichten werden an der KDCS-Schnittstelle bei den Aufrufen zur Nachrichtenbehandlung (MPUT/FPUT/DPUT) im Nachrichtenbereich übergeben.

'U' (USER)
UTM konvertiert die Benutzer-Nachrichten nicht, d.h. die Nachrichten werden unverändert zwischen den Partner-Anwendungen übertragen.

'1', '2', '3', '4' (SYS1 | SYS2 | SYS3 | SYS4)

ist nur für folgende TS-Anwendungen möglich:

- BS2000-Systeme: *ptype*='SOCKET'
- Unix-, Linux- oder Windows-Systeme: *ptype*='APPLI' oder 'SOCKET'

UTM konvertiert die Benutzernachrichten gemäß den für die Code-Konvertierung bereitgestellten Konvertierungstabellen, siehe Abschnitt „Code-Konvertierung“ im openUTM-Handbuch „Anwendungen generieren“, d.h.:

- Vor dem Senden wird auf Unix-, Linux- und Windows-Systemen von ASCII nach EBCDIC und auf BS2000-Systemen von EBCDIC nach ASCII konvertiert.

- Nach dem Empfangen wird auf Unix-, Linux- und Windows-Systemen von EBCDIC nach ASCII und auf BS2000-Systemen von ASCII nach EBCDIC konvertiert.

Dabei geht UTM davon aus, dass die Nachricht nur abdruckbare Zeichen enthält.

Weitere Informationen zur Code-Konvertierung finden Sie im openUTM-Handbuch „Anwendungen programmieren mit KDCS“, Abschnitt "Code-Konvertierung".

termn (**ter**minal **mn**emonic)

enthält das Kennzeichen für die Art des Kommunikationspartners. Das Kennzeichen stellt UTM KDCS-Teilprogrammen im Feld KCTERMN des KB-Kopfes zur Verfügung.
Die Standardwerte für *termn* können Sie der Tabelle bei *ptype* ("BS2000-Systeme, Unix-, Linux-" und "Windows-Systeme") entnehmen.

protocol

Nur auf BS2000-Systemen: gibt an, ob auf Verbindungen zwischen der UTM-Anwendung und dem Client /Drucker mit dem Benutzerdienstprotokoll NEABT ("Stationsprotokoll") gearbeitet wird.

'N' Zwischen der UTM-Anwendung und dem Client/Drucker wird ohne Benutzerdienstprotokoll gearbeitet.

Es wird immer *protocol*='N' ausgegeben für:

- UPIC-Clients (*ptype*='UPIC-R')
- TS-Anwendungen (*ptype*='APPLI' oder 'SOCKET')
- Drucker, die über RSO angesprochen werden (*ptype*='RSO')

Clients, für die *protocol*='N'gesetzt ist, können sich nicht über eine Multiplexverbindung an die Anwendung anschließen.

'S' (STATION)
Zwischen der UTM-Anwendung und dem Client/Drucker wird mit dem Benutzerdienstprotokoll (NEABT) gearbeitet.

UTM verwendet das Benutzerdienstprotokoll NEABT u.a. zur Bestimmung des Typs (*ptype*) eines Clients oder Druckers, wenn der Typ beim Eintragen des Client/Druckers nicht explizit angegeben wurde (eingetragen mit *ptype*='ANY').

usage_type

Nur auf BS2000-Systemen: gibt an, ob der in *pt_name* angegebene Kommunikationspartner ein Dialog-Partner oder ein reines Ausgabemedium ist.

'D' Der Client ist ein Dialog-Partner. Auf den Verbindungen zwischen Client und lokaler Anwendung kann sowohl der Client als auch die lokale Anwendung Nachrichten senden.
UPIC-Clients sind immer Dialog-Partner (*ptype*='UPIC-R').

'O' Der Client/Drucker wird als reines Ausgabemedium genutzt. Es können nur Nachrichten von der Anwendung zum Client/Drucker gesendet werden. Für Drucker wird immer *usage_type*='O' ausgegeben.

listener_port

- auf BS2000-Systemen nur relevant, wenn *t_prot*='T'.
- auf Unix-, Linux- und Windows-Systemen nur relevant, wenn *t_prot*='T' oder 'R'.

listener_port enthält die Portnummer der Partner-Anwendung im fernen System. Bei KC_GET_OBJECT wird die Portnummer zurückgeliefert, die bei der Generierung des Client angegeben wurde.

listener_port='0' bedeutet, dass bei der Generierung des Client keine Portnummer angegeben wurde.

t_prot

enthält das Adressformat, mit dem sich der Client beim Transportsystem anmeldet.

Folgende Adressformate sind möglich:

'T' native TCP/IP-Transportprotokoll für die Kommunikation über die Socket-Schnittstelle (SOCKET)

Nur auf Unix-, Linux- und Windows-Systemen:

'R' RFC1006, ISO-Transportprotokoll Klasse 0 über TCP/IP und Konvergenzprotokoll RFC1006.

Wurde dem Client beim Eintragen in die Konfiguration kein Adressformat zugeordnet, dann enthält das Feld *t_prot* ein Leerzeichen.

tsel_format

Nur auf Unix-, Linux- und Windows-Systemen: enthält den Formatindikator des T-Selektors in der Adresse des Client.

Folgende Formatindikatoren können auftreten:

'T' TRANSDATA-Format

'E' EBCDIC-Zeichenformat

'A' ASCII-Zeichenformat

Zur Bedeutung der Adressformate siehe „Dokumentation zu PCMX“ im Abschnitt "openUTM-Dokumentation".

Wurde dem Client beim Eintragen in die Konfiguration kein Formatindikator zugeordnet, dann enthält das Feld *tsel_format* ein Leerzeichen.

connect_mode

gibt den aktuellen Zustand der Verbindung zum Client an:

'Y' Die Verbindung ist aufgebaut.

'W' UTM versucht gerade, die Verbindung aufzubauen (waiting for connection).

'N' Die Verbindung ist nicht aufgebaut.

In UTM-Anwendungen auf BS2000-Systemen kann connect_mode für Clients/ Drucker, die über einen Multiplexanschluss mit der Anwendung verbunden sind (mux='Y'), auch die folgenden Werte enthalten:

'T'

(timer)
Die Session zum Client ist im Zustand DISCONNECT-PENDING; der Timer für das Warten auf die Bestätigung des Verbindungsabbaus läuft.

'E' (expired)
Die Session ist im Zustand DISCONNECT-PENDING und der Timer für das Warten auf die Bestätigung ist ohne Eintreffen der Bestätigung abgelaufen.
Die Session muss explizit freigegeben werden (z.B. mit KC_MODIFY_OBJECT und *connect_mode* ='R', siehe Abschnitt "obj_type=KC_PTERM").

pool

gibt an, ob der Client über einen LTERM-Pool mit der Anwendung verbunden ist.

'Y' Der Client ist über einen LTERM-Pool mit der Anwendung verbunden.

'N' Der Client ist nicht über einen LTERM-Pool mit der Anwendung verbunden.

mux (Nur auf BS2000-Systemen)

gibt an, ob der Client über einen Multiplexanschluss mit der Anwendung verbunden ist.

'Y' Der Client ist über einen Multiplexanschluss mit der Anwendung verbunden.

'N' Der Client ist nicht über einen Multiplexanschluss mit der Anwendung verbunden.

contime_min

gibt an, wie lange die Verbindung zum Client/Drucker bereits besteht. Angegeben wird die Dauer in Minuten.

letters

enthält die Anzahl der Ein- und Ausgabe-Nachrichten für den Client/Drucker seit dem letzten Start der lokalen Anwendung.

conbad

gibt an, wie oft die Verbindung zum Client/Drucker seit dem letzten Start der Anwendung ausgefallen ist.

deleted

gibt an, ob der Client/Drucker aus der Konfiguration gelöscht wurde oder nicht.

'Y' Der Client/Drucker wurde gelöscht. Der Name ist aber gesperrt, d.h. es kann kein neuer Client /Drucker mit diesem Namen eingetragen werden.

'N' Der Client/Drucker wurde nicht gelöscht.

idletime

nur relevant bei Dialog-Partnern.

idletime enthält die Zeit in Sekunden, die UTM nach dem Ende einer Transaktion oder nach dem Anmelden maximal auf eine Eingabe vom Client wartet. Bei Zeitüberschreitung wird die Verbindung zum Client abgebaut. Ist der Client ein Terminal, dann wird vor dem Verbindungsabbau die Meldung K021 ausgegeben. 0 bedeutet Warten ohne Zeitbegrenzung

encryption_level

nur relevant für UPIC-Clients und auf BS2000-Systemen für einige Terminalemulationen.
encryption_level gibt an, ob die UTM-Anwendung auf der Verbindung zum Client

- die Verschlüsselung der Nachrichten standardmäßig anfordert oder nicht,
- welche Verschlüsselungsebene dabei gefordert wird,
- oder ob der Client ein „trusted“ Client ist.

Folgende Werte sind möglich:

'N' (NONE)
UTM fordert **keine** Verschlüsselung der Nachrichten an, die zwischen dem Client und der UTM-Anwendung ausgetauscht werden.
Services, für die die Verschlüsselung generiert wurde (siehe *kc_tac_str.encryption_level* im Abschnitt "kc_tac_str - Transaktionscodes lokaler Services"), können von diesem Client nur gestartet werden, wenn der Client explizit die Verschlüsselung auswählt.
Passworte werden verschlüsselt übertragen, sofern beide Partner Verschlüsselung unterstützen.

'3' (LEVEL 3)
UTM fordert die Verschlüsselung der Nachrichten mit der Verschlüsselungsebene 3 an, d.h. die Nachrichten werden mit dem AES-CBC Algorithmus verschlüsselt und zum Austausch des AES-Schlüssels wird ein RSA Schlüssel mit einer Schlüssellänge von 1024 bit verwendet.
Der Verbindungsaufbau zu dem Client wird von UTM abgelehnt, wenn der Client nicht mindestens diese Verschlüsselungsebene unterstützt.

'4' (LEVEL 4)
UTM fordert die Verschlüsselung der Nachrichten mit der Verschlüsselungsebene 4 an, d.h. die Nachrichten werden mit dem AES-CBC Algorithmus verschlüsselt und zum Austausch des AES-Schlüssels wird ein RSA Schlüssel mit einer Schlüssellänge von 2048 bit verwendet.
Der Verbindungsaufbau zu dem Client wird von UTM abgelehnt, wenn der Client nicht mindestens diese Verschlüsselungsebene unterstützt.

'5' (LEVEL 5) nur auf Unix-, Linux- und Windows-Systemen
Schlüssellänge wie bei LEVEL 4. Zur Vereinbarung des Session-Keys wird das auf Elliptic Curves basierende Diffie-Hellman Verfahren verwendet und Ein-/Ausgabe-Nachrichten werden mit dem AES-GCM Algorithmus verschlüsselt. Der Verbindungsaufbau zu dem Client wird von UTM abgelehnt, wenn der Client nicht mindestens diese Verschlüsselungsebene unterstützt.

'T' (TRUSTED)
Der Client ist ein „trusted“ Client.
Nachrichten und Passworte zwischen Client und Anwendung werden nicht verschlüsselt.
Ein „trusted“ Client kann auch Vorgänge starten, deren Vorgangs-TACs Verschlüsselung erfordern (generiert mit *kc_tac_str.encryption_level* ='1' oder '2'; siehe Abschnitt "kc_tac_str -

Transaktionscodes lokaler Services").
Verbindungen, die über einen Transportsystemendpunkt (BCAMAPPL) aufgebaut werden, der mit T-PROT=(..., SECURE) generiert ist, werden von UTM immer als trusted eingestuft.

ip_addr

liefert die von UTM für diese Verbindung verwendete IP-Adresse des Partners, wenn es sich um eine IPv4-Adresse handelt.

Auf BS2000-Systemen werden IP-Adressen nur für Partner mit *ptype*='SOCKET' ausgegeben.

Eine IPv6-Adresse wird im Feld *ip_addr_v6* zurückgeliefert (siehe Abschnitt "kc_pterm_str - Clients und Drucker").

In *ip_addr* liefert UTM die IP-Adresse des Client-Rechners zurück, die in der Objekttabelle der Anwendung abgelegt ist. Diese Adresse benutzt UTM, um die Verbindung zum Client aufzubauen. Die IP-Adresse wird von UTM mit Hilfe des generierten Rechnernamens (*pronam*) beim Anwendungsstart aus dem Name-Service gelesen.

Existiert in den Objekttabellen für den Client keine IPv4-Adresse, dann liefert UTM in *ip_addr* Leerzeichen zurück.

curr_encryption

nur relevant für UPIC-Clients und auf BS2000-Systemen für einige Terminalemulationen.

Für eine bestehende Verbindung zum Client liefert UTM in *curr_encryption* die Verschlüsselungsebene zurück, die zwischen UTM-Anwendung und Client für diese Verbindung ausgehandelt wurde. Zu den Eigenschaften der Verschlüsselungsebenen 3 bis 5 siehe Abschnitt "kc_pterm_str - Clients und Drucker".

'N' (NONE)
Die Nachrichten, die auf der Verbindung ausgetauscht werden, werden nicht verschlüsselt. Passworte werden verschlüsselt übertragen, sofern beide Partner Verschlüsselung unterstützen.

'3' (LEVEL 3)
Auf der Verbindung werden alle Nachrichten verschlüsselt übertragen.
Dabei wird die Verschlüsselungsebene 3 verwendet.

'4' (LEVEL 4)
Auf der Verbindung werden alle Nachrichten verschlüsselt übertragen.
Dabei wird die Verschlüsselungsebene 4 verwendet.

'5' (LEVEL 5) (nur auf Unix-, Linux- und Windows-Systemen)
Auf der Verbindung werden alle Nachrichten verschlüsselt übertragen.
Dabei wird die Verschlüsselungsebene 5 verwendet.

' ' (Leerzeichen)
Zu diesem Client existiert zur Zeit keine Verbindung.

usp_hdr

nur relevant für Socket-Partner.
Zeigt an, für welche Ausgabe-Nachrichten UTM auf dieser Verbindung einen UTM-Socket-Protokoll-Header aufbaut. Mögliche Werte sind:

'A' Bei allen Ausgabe-Nachrichten (Dialog, Asynchron, K-Meldungen) erzeugt UTM einen UTM-Socket-Protokoll-Header und stellt diesen der Nachricht voran.

'M' Nur bei Ausgabe von K-Meldungen erzeugt UTM einen UTM-Socket-Protokoll-Header und stellt diesen der Nachricht voran.

'N' UTM erzeugt für keine Ausgabe-Nachricht einen UTM-Socket-Protokoll-Header.

ip_addr_v6

liefert die von UTM für diese Verbindung verwendete IP-Adresse des Partners, wenn es sich um eine IPv6-Adresse handelt oder um eine IPv4-Adresse, die in ein IPv6-Format eingebettet ist.

Auf BS2000-Systemen werden IP-Adressen nur für Partner mit *ptype*='SOCKET' ausgegeben.

Eine IPv4-Adresse wird im Feld *ip_addr* zurückgeliefert (siehe Abschnitt "kc_pterm_str - Clients und Drucker").

In *ip_addr_v6* liefert UTM die IP-Adresse des Client-Rechners zurück, die in der Objekttabelle der Anwendung abgelegt ist. Diese Adresse benutzt UTM, um die Verbindung zum Client aufzubauen. Die IP-Adresse wird von UTM mit Hilfe des generierten Rechnernamens (*pronam*) beim Anwendungsstart aus dem Name-Service gelesen.

Existiert in den Objekttabellen für den Client keine IPv6-Adresse, dann liefert UTMin *ip_addr_v6* Leerzeichen zurück.

ip_v

gibt an, ob es sich bei der von UTM für diese Verbindung verwendeten IP-Adresse um eine IPv4- oder um eine IPv6-Adresse handelt:

'V4' IPv4-Adresse.

'V6' IPv6-Adresse oder IPv4-Adresse, die in ein IPv6-Format eingebettet ist.

Wenn keine IP-Adresse geliefert werden kann, liefert openUTM Leerzeichen zurück.

pronam_long

Name des Rechners, auf dem sich der Client bzw. an dem sich der Drucker befindet.

In UTM-Anwendungen auf BS2000-Systemen ist *pronam_long* immer belegt. Der Rechnername in *pronam_long* ist identisch mit dem Namen, der bei der BCAM-Generierung bzw. in der RTF-Datei für diesen Rechner festgelegt wurde. Bei einem RSO-Drucker enthält *pronam_long* den Wert '*RSO'.

In UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen enthält *pronam_long* bei einem lokalen Client oder einem Drucker Leerzeichen.

11.3.1.26 kc_queue_str - Eigenschaften temporärer Queues

Für den Objekttyp KC_QUEUE ist die Datenstruktur *kc_queue_str* definiert.
Bei KC_GET_OBJECT liefert UTM in *kc_queue_str* Informationen über die in der Anwendung existierenden temporären Queues zurück.

Datenstruktur kc_queue_str
`char qu_name[8];`
`char qlev[5];`
`char queue_length[8];`
`char q_mode;`

Die Felder der Datenstruktur haben die folgende Bedeutung:

qu_name

Name, der beim Erzeugen der Queue mit QCRE vereinbart oder von UTM automatisch vergeben wurde.

qlev

enthält die maximale Anzahl von Nachrichten, die gleichzeitig in der Queue stehen dürfen.

UTM berücksichtigt die für die Queue erzeugten Nachrichten erst am Ende der Transaktion. Daher kann die in *qlev* festgelegte Anzahl von Nachrichten für eine Message Queue überschritten werden, wenn in einer Transaktion mehrere Nachrichten für dieselbe Queue erzeugt wurden.

qlev=32767 bedeutet, dass die Anzahl der Nachrichten in der Queue nicht beschränkt ist.

queue_length

enthält die Anzahl der Nachrichten in der Queue, die gerade bearbeitet werden oder auf Bearbeitung warten.

q_mode

zeigt an, wie sich UTM verhält, wenn die maximale Anzahl der erlaubten Nachrichten für die Queue erreicht ist. Mögliche Werte sind:

'S' (STD)
UTM lehnt weitere Nachrichten für diese Queue ab.

'W' (WRAP-AROUND)
UTM nimmt weitere Nachrichten auf. Beim Eintrag einer neuen Nachricht wird die älteste in der Queue stehende Nachricht gelöscht.

11.3.1.27 kc_sfunc_str - Funktionstasten

Für den Objekttyp KC_SFUNC ist die Datenstruktur *kc_sfunc_str* definiert. In *kc_sfunc_str* liefert UTM bei KC_GET_OBJECT die Kurzbezeichnung einer in der Anwendung generierten Funktionstaste zurück und gibt an, welche Funktion dieser Funktionstaste zugeordnet ist.

Einer Funktionstaste kann entweder ein Transaktionscode, ein Kommando, ein KDCS-Returncode zugeordnet sein oder sie kann zur Kellerung von Vorgängen verwendet werden.

Für UPIC-Clients wird nur der Parameter *ret* ausgewertet.

Datenstruktur kc_sfunc_str
`char sf_name[4];`
`char tac[8];`
`char stack[8];`
`char ret[3];`
`char cmd[8];`

Die Felder der Datenstruktur haben die folgende Bedeutung:

sf_name

enthält die Kurzbezeichnung für die Funktionstaste. Mögliche Werte sind:

- BS2000-Systeme: K1 bis K14 und F1 bis F24

  Mit K-Tasten werden Kurznachrichten ausgelöst, bei denen nur der Wert der K-Taste übermittelt wird. K14 wird für Ausweisleser benötigt (siehe openUTM-Handbuch „Anwendungen programmieren mit KDCS“, Ausweisleser).
- Unix-, Linux- und Windows-Systeme: F1 bis F20

Mit F-Tasten können der Wert der F-Taste und eine Eingabe-Nachricht übermittelt werden.

tac enthält den Namen des Transaktionscodes (Vorgangs-TAC), der dieser Funktionstaste zugeordnet ist.

Wird die Funktionstaste außerhalb eines Vorgangs gedrückt, dann wird der zu dem Transaktionscode gehörende Vorgang gestartet.

Wird die Funktionstaste innerhalb eines Vorgangs gedrückt, dann wird die Funktion wirksam, die der Funktionstaste über *ret* oder *stack* zugeordnet ist. Sind diese beiden Felder nicht belegt, dann liefert der erste MGET-Aufruf im nächsten Teilprogramm des aktuellen Vorgangs den Returncode 19Z.

stack wird zur Kellerung von Vorgängen verwendet. *stack* enthält den Namen des Dialog-Transaktionscodes, der dieser Funktionstaste zugeordnet ist.

Wird die Funktionstaste innerhalb eines Vorgangs gedrückt, dann wird der aktuelle Vorgang gekellert und der Vorgang mit dem Transaktionscode in *stack* gestartet.

Wird die Funktionstaste außerhalb eines Vorgangs gedrückt, dann wird der Transaktionscode gestartet, der im Feld *tac* steht. Ist das Feld *tac* leer, dann bewirkt das Drücken der Funktionstaste das Starten des Vorgangs mit dem in *stack* angegebenen Transaktionscode.

ret enthält einen KDCS-Returncode.

Wird diese Funktionstaste während eines Vorgangs gedrückt, dann steht dieser Returncode nach dem MGET-Aufruf im Feld KCRCCC des Kommunikationsbereiches.

Wird diese Funktionstaste zu Vorgangsbeginn gedrückt und ist *tac* nicht belegt, dann reagiert UTM mit der Meldung K009 oder startet das BADTACS-Teilprogramm. Dieses erhält beim ersten MGET-Aufruf den der Funktionstaste zugeordneten Returncode im Feld KCRCCC.

Mögliche Werte: 20Z <= *ret* <= 39Z.

Wird die Funktionstaste von einem UPIC-Client übermittelt, dann wird nur das Feld *ret* ausgewertet.

cmd Name eines KDC-Kommandos (z.B. KDCOFF oder ein Administrationskommando wie z.B. KDCINF), das ausgeführt wird, wenn die Funktionstaste gedrückt wird.

Ist *cmd* mit einem Wert belegt, dann sind die Felder *tac*, *stack* und *ret* mit Leerzeichen belegt.

11.3.1.28 kc_subnet_str - Information zu Subnetzen

Für den Objekttyp KC_SUBNET ist die Datenstruktur *kc_subnet_str* definiert. UTM liefert bei KC_GET_OBJECT die Generierungsdaten zu Subnetzen zurück.

Datenstruktur kc_subnet_str
`char mapped_name[8];`
`char relevant_bits[3];`
`char ip_addr_format[2];`
`char ipv4_address[15];`
`char ipv6_address[39];`
`char bcamappl[8];`
`char resolve_names;`

Die Felder der Datenstruktur haben die folgende Bedeutung:

mapped_name

enthält den *mapped_name* aus der KDCDEF-Anweisung SUBNET.

relevant_bits

enthält die Anzahl der für die Subnetzmaske relevanten Bits.

ip_addr_format

gibt das Adressformat an:

V4 es handelt sich um eine IPv4-Subnetzadresse.

V6 es handelt sich um eine IPv6-Subnetzadresse.

ipv4_address

enthält bei *ip_addr_format*=V4 die IPv4-Subnetzadresse, andernfalls Leerzeichen.

ipv6_address

enthält bei *ip_addr_format*=V6 die IPv6-Subnetzadresse, andernfalls Leerzeichen.

bcamappl

enthält den BCAMAPPL-Namen aus der KDCDEF-Anweisung SUBNET.

resolve_names

gibt an, ob nach einem Verbindungsaufbau eine Namensauflösung per DNS stattfinden soll oder nicht. Siehe KDCDEF-Anweisung SUBNET.

11.3.1.29 kc_tac_str - Transaktionscodes lokaler Services

Für den Objekttyp KC_TAC ist die Datenstruktur *kc_tac_str* definiert. Bei KC_GET_OBJECT liefert UTM in *kc_tac_str* die folgenden Informationen zurück:

- Eigenschaften eines Transaktionscodes oder einer TAC-Queue.
- Statistikinformationen über die Auslastung des Services.
- aktueller Zustand des Transaktionscodes oder der TAC-Queue.

Für die Auswertung der Informationen von TAC-Queues (*tac_type*='Q') sind nur die Felder *tc_name*, *admin*, *qlev*, *q_mode*, *q_read_acl*, *q_write_acl* und *state* von Bedeutung.

Transaktionscodes können dynamisch mit KC_CREATE_OBJECT erzeugt, mit KC_DELETE_OBJECT gelöscht oder mit KC_MODIFY_OBJECT modifiziert werden.

mod[1]	**Datenstruktur kc_tac_str**
-	`char tc_name[8];`
-	`char program[32];`
x(GPD)	`char lock_code[4];`
x(GID)	`char state;`
-	`char tacclass[2];`
-	`char admin;`
-	`char call_type;`
-	`char exit_name[32];`
-	`char qlev[5];`
-	`char tac_type;`
-	`char real_time_sec[5];`
-	`char cpu_time_msec[8];` (nur auf BS2000-Systemen)
-	`char dbkey[8];` (nur auf BS2000-Systemen)
-	`char runprio[3];` (nur auf BS2000-Systemen)
-	`char api;`
-	`char satadm;` (nur auf BS2000-Systemen)
-	`char satsel;` (nur auf BS2000-Systemen)
-	`char tacunit[4];`

mod[1]	Datenstruktur kc_tac_str
-	`char tcbentry[8];` (nur auf BS2000-Systemen)
-	`char in_queue[5];`
x(GIR)	`char used[10];`
x(GIR)	`char number_errors[5];`
x(GIR)	`char db_counter[10];`
x(GIR)	`char tac_elap_msec[10];`
x(GIR)	`char db_elap_msec[10];`
x(GIR)	`char taccpu_msec[10];`
-	`char deleted;`
-	`char pgwt;`
-	`char encryption_level;`
x(GPD)	`char access_list[8];`
-	`char q_mode;`
x(GPD)	`char q_read_acl[8];`
x(GPD)	`char q_write_acl[8];`
-	`char nbr_dputs[10];`
-	`char nbr_ack_jobs[10];`
x(GPD)	`char dead_letter_q;`
x(GIR)	`char nbr_ta_commits[10];`
x(GIR)	`char number_errors_ex[10];`
-	`char in_queue_ex[10];`
x(GIR)	`char taccpu_micro_sec[10];`

[1] Feldinhalt mit KC_MODIFY_OBJECT modifizierbar; siehe Abschnitt "obj_type=KC_TAC"

Die Felder der Datenstruktur haben die folgende Bedeutung:

tc_name

enthält den Namen des Transaktionscodes bzw. der TAC-Queue, deren Eigenschaften UTM zurückliefert. Der Name ist bis zu 8 Zeichen lang.

program

enthält den Namen des Teilprogramms, dem dieser Transaktionscode zugeordnet ist.

Für TAC-Queues werden Leerzeichen zurückgegeben.

lock_code

enthält den Lockcode, der dem Transaktionscode zugeordnet ist (Zugriffsschutz). Nur Benutzer/Clients, die den entsprechenden Keycode besitzen, dürfen den Transaktionscode aufrufen. Der Keycode muss sowohl im Keyset der Benutzerkennung als auch im Keyset des LTERM-Partners enthalten sein, über den sich der Benutzer/Client an die Anwendung anschließt.

lock_code kann eine Zahl zwischen '0' und '4000' enthalten.
In KC_CREATE_OBJECT darf *lock_code* höchstens den mit dem Operanden KEYVALUE der KDCDEF-Anweisung MAX definierten Maximalwert enthalten.

'0' bedeutet, dass es keinen Lockcode gibt; d.h. der Transaktionscode nicht durch einen Lockcode geschützt ist.

Wenn Sie den Lockcode verändern wollen, darf im Feld *access_list* kein Keyset eingetragen sein.

Für TAC-Queues ist dieser Parameter nicht erlaubt. In diesem Fall werden Leerzeichen zurückgegeben.

state

gibt an, ob der TAC oder die TAC-Queue freigegeben oder gesperrt ist:

'Y' TACs: Der Transaktionscode ist nicht gesperrt. Er ist nach dem Start der Anwendung verfügbar. Er ist solange verfügbar, bis er explizit gesperrt oder gelöscht wird.

TAC-Queues: Schreiben und Lesen ist erlaubt.

'N' TACs: Der Transaktionscode ist gesperrt. Die Sperre *state*='N' bedeutet, dass UTM keine Aufträge für diesen TAC mehr annimmt.
Handelt es sich um den Transaktionscode eines KDCS-Teilprogramms mit *call_type*='B', dann ist der Transaktionscode als Vorgangs-TAC (1. TAC eines Vorgangs) gesperrt. Als Folge-TAC in einem Vorgang ist er jedoch nicht gesperrt. Folge-TACs (*call_type*='N') können nicht mit *state*='N' gesperrt werden.

TAC-Queues: Schreibzugriffe sind gesperrt, Lesen ist erlaubt.

'H' (HALT)
TACs: Der Transaktionscode ist vollständig gesperrt. Das zugehörige Teilprogramm wird von UTM nicht mehr gestartet, wenn es mit diesem Transaktionscode aufgerufen wird. Für den Transaktionscode eines KDCS-Teilprogramms heißt das, dass er als Vorgangs-TAC und auch als Folge-TAC in einem Asynchron- oder Dialog-Vorgang gesperrt ist.
Ist der Transaktionscode ein Vorgangs-TAC, dann werden für ihn keine Aufträge angenommen.
Wird dieser TAC in einem Vorgang als Folge-TAC aufgerufen, dann wird der Vorgang abgebrochen (interner PEND ER mit 74Z).

Asynchron-Aufträge, die bereits vor der Sperre in der Message Queue des TACs zwischengespeichert wurden, werden nicht gestartet. Sie bleiben in der Queue bis der TAC wieder freigegeben bzw. auf *state*='N' gesetzt wird.

TAC-Queues: Schreiben und Lesen ist nicht möglich.

'K'

(KEEP)
TACs: kann nur bei Asynchron-Transaktionscodes, die auch Vorgangs-TACs (*call_type* ='B'oder 'F') sind, auftreten.

UTM nimmt Aufträge für den Transaktionscode an. Die Aufträge werden jedoch nicht bearbeitet, sondern lediglich in die Auftragswarteschlange des Transaktionscodes geschrieben. Sie werden bearbeitet, wenn Sie den Status des Transaktionscodes ändern in 'Y' oder 'N'.

TAC-Queues: Schreiben ist erlaubt aber Lesen ist nicht möglich.

Sie können einen Transaktionscode oder eine TAC-Queue im laufenden Betrieb sperren bzw. freigeben.

tacclass

enthält die TAC-Klasse, der der Transaktionscode zugeordnet ist. *tacclass* enthält eine Zahl zwischen 1 und 16 oder Leerzeichen. Dabei sind:

1 - 8 Dialog-TAC-Klassen

9 - 16 Asynchron-TAC-Klassen

Gibt UTM in *tacclass* Leerzeichen zurück, gilt:

- Bei der KDCDEF-Generierung wurden keine TAC-Klassen erzeugt oder
- Der Transaktionscode in *tc_name* ist ein Dialog-TAC (*tac_type*='D'), der keiner TAC-Klasse zugeordnet ist.

admin

gibt bei *tac_type*='A' oder 'D' an, welche Berechtigung ein Benutzer oder Client benötigt, um diesen Transaktionscode bzw. einen Vorgang aufzurufen, der diesen Transaktionscode als Folge-TAC enthält. Bei *tac_type*='Q' zeigt *admin* an, welche Berechtigung ein Benutzer oder Client braucht, um auf diese TAC-Queue zuzugreifen.

'Y' TACs: Diesen Transaktionscode kann nur ein Benutzer mit Administrationsberechtigung aufrufen.

TAC-Queues: Nur ein Benutzer mit Administrationsberechtigung kann Nachrichten in diese Queue schreiben bzw. aus dieser Queue lesen.

'N' Für diesen TAC bzw. diese TAC-Queue ist keine Administrationsberechtigung erforderlich.

'R' (READ)
Für diesen TAC bzw. diese TAC-Queue ist keine Administrationsberechtigung erforderlich.

Das zu dem Transaktionscode gehörende Teilprogramm darf alle Funktionen von KDCADMI nutzen, die lesend auf die Anwendungsdaten zugreifen.

Darüber hinaus können die Zugriffsrechte auf den TAC (*tac_type*='A' oder 'D') durch einen Lockcode oder eine Access Liste eingeschränkt sein. Handelt es sich um eine TAC-Queue (*tac_type*='Q'), ist eine Einschränkung der Zugriffsrechte über die Parameter *q_read_acl* und/oder *q_write_acl* möglich.

call_type

gibt an, ob mit dem Transaktionscode ein Service gestartet wird (z.B. 1. TAC eines Vorgangs) oder ob er Folge-TAC in einem Vorgang ist.

'B' (BOTH)
Mit dem TAC kann ein Service gestartet werden. Er kann aber auch Folge-TAC in einem Vorgang sein.

'F' (FIRST)
Mit dem Transaktionscode kann ein Service gestartet werden.

'N' (NEXT)
Der Transaktionscode kann nur Folge-TAC in einem Vorgang sein. Er kann nur mit *state*='H' gesperrt werden.

exit_name

enthält den Namen des Event-Exit VORGANG, der diesem TAC zugeordnet ist.

qlev (**q**ueue **lev**el)

qlev gibt bei Asynchron-Transaktionscodes (*tac_type*='A') oder bei Queues (*tac_type*='Q') an, wieviele Nachrichten maximal in der Message Queue für diesen Transaktionscode bzw. in der TAC-Queue stehen dürfen. Wird dieser Schwellwert überschritten, hängt das weitere Verhalten von UTM vom Wert im Feld *q_mode* ab.

UTM berücksichtigt die für die Queue erzeugten Nachrichten erst am Ende der Transaktion. Daher kann die in *qlev* festgelegte Anzahl von Nachrichten für eine Message Queue überschritten werden, wenn in einer Transaktion mehrere Nachrichten für dieselbe Queue erzeugt wurden.

tac_type

gibt an, ob Aufträge an diesen Transaktionscode im Dialog oder asynchron bearbeitet werden oder ob eine TAC-Queue generiert wurde.

'D' Dieser Transaktionscode ist ein Dialog-TAC. Aufträge an diesen Transaktionscode werden im Dialog mit dem Auftraggeber bearbeitet.

'A' Dieser Transaktionscode ist ein Asynchron-Transaktionscode. Beim Aufruf dieses Transaktionscodes wird ein Asynchron-Auftrag erzeugt, der in der Message Queue des Transaktionscodes zwischengespeichert wird. Die Bearbeitung des Auftrags erfolgt entkoppelt vom Auftraggeber.

'Q' Es wurde eine TAC-Queue generiert.

In eine solche Queue kann mit einem DPUT-Aufruf eine Nachricht geschrieben und aus der Queue kann mit einem DGET-Aufruf gelesen werden.

real_time_sec

enthält die Realzeit in Sekunden, die ein Teilprogramm maximal verbrauchen darf, wenn es über diesen Transaktionscode gestartet wird. Läuft das Teilprogramm länger, dann bricht UTM den Vorgang ab und gibt eine Meldung aus. *real_time_sec*='0' bedeutet, dass der Realzeit-Verbrauch des Teilprogramms nicht überwacht wird.

cpu_time_msec (nur auf BS2000-Systemen)

enthält die CPU-Zeit in Millisekunden, die das Teilprogramm mit diesem Transaktionscode während einer Verarbeitung maximal verbrauchen darf. Läuft das Teilprogramm länger, bricht UTM den Vorgang mit einer Meldung ab.
Der Wert '0' bedeutet, dass für das Teilprogramm, das über diesen Transaktionscode gestartet wird, keine Zeitüberwachung erfolgen soll.

dbkey (nur auf BS2000-Systemen)

ist nur relevant, wenn das zum Transaktionscode gehörende Teilprogramm Datenbankaufrufe absetzt und das Datenbanksystem mit UTM gekoppelt ist. *dbkey* enthält den Datenbankschlüssel, den UTM bei einem Datenbankaufruf aus dem Teilprogramm an das Datenbanksystem übergibt. Das Format des Schlüssels ist abhängig vom verwendeten Datenbanksystem. Der Schlüssel ist maximal 8 Zeichen lang.
Zur Zeit wird *dbkey* nur für UDS unterstützt.

Der Wert *dbkey*='UTM' bewirkt, dass der Wert des Startparameters DBKEY an die Datenbank übergeben wird (siehe openUTM-Handbuch „Einsatz von UTM-Anwendungen auf BS2000-Systemen“; Startparameter).

runprio (nur auf BS2000-Systemen)

enthält die für den Transaktionscode eingestellte Run Priorität des BS2000-Systems. Diese Run Priorität wird dem UTM-Prozess zugeordnet, in dem das zugehörige Teilprogramm abläuft. So können Sie die Scheduling Mechanismen des BS2000-Systems zur Ablaufsteuerung von UTM-Teilprogrammläufen einsetzen. Die Run Priorität hat jedoch keinen Einfluss auf den Zeitpunkt, zu dem UTM ein Teilprogramm startet.

Beim Start eines Teilprogramms versucht UTM, die Run Priorität des aktuellen Prozesses auf den Wert in *runprio* zu setzen. Ist die generierte Run Priorität nicht mit den JOIN-Einträgen der entsprechenden Benutzerkennung verträglich, dann wird die Run Priorität des aktuellen Prozesses nicht geändert. UTM gibt eine entsprechende K-Meldung aus. Sind die maximal erlaubten *runprio*-Werte für die Benutzerkennung und die Jobklasse unterschiedlich, so wird der für den Benutzer günstigere Wert erlaubt. Sind keine JOIN-Einträge vorhanden, wird die in *runprio* angegebene Run Priorität gesetzt.

Nach Beendigung eines Teilprogrammlaufs setzt UTM die Run Priorität wieder auf den ursprünglich eingestellten Wert zurück, es sei denn, die Run Priorität wurde während des Teilprogrammlaufs mit dem CHANGE-TASK-PRIORITY-Kommando nochmals geändert. In diesem Fall wird die von außen eingestellte Run Priorität nach Teilprogrammende beibehalten.

Ist *runprio*='0', dann ist für diesen Transaktionscode keine spezifische Run Priorität generiert.

api (**a**pplication **p**rogramming **i**nterface)

gibt an, welche Programmschnittstelle das zum Transaktionscode gehörende Teilprogramm verwendet.

'K' KDCS

'C' CPI-C

'X' XATMI

satadm (nur auf BS2000-Systemen)

gibt an, ob zum Aufrufen des Transaktionscodes die UTM-SAT-Administrationsberechtigung erforderlich ist.

'Y' Der TAC darf nur von Benutzern, Clients bzw. Partner-Anwendungen aufgerufen werden, die zur Administration der SAT-Protokollierung innerhalb der Anwendung berechtigt sind (UTM-SAT-Administrationsberechtigung).

'N' Der Transaktionscode darf auch von Benutzern, Clients und Partner-Anwendungen aufgerufen werden, die keine UTM-SAT-Administrationsberechtigung haben.

satsel (nur auf BS2000-Systemen)

gibt an, welche Ereignisse von SAT beim Ablauf des zugehörigen Teilprogramms protokolliert werden (TAC-spezifische Einstellung). Voraussetzung für die Protokollierung ist, dass die SAT-Protokollierung für die Anwendung eingeschaltet ist (*kc_max_par_str.sat*='Y'). Zur SAT-Protokollierung siehe auch openUTM-Handbuch „Anwendungen generieren“ und openUTM-Handbuch „Einsatz von UTM-Anwendungen auf BS2000-Systemen“.

'B' (BOTH)
Es werden erfolgreiche **und** nicht erfolgreiche Ereignisse protokolliert.

'S' (SUCCESS)
Es werden nur erfolgreiche Ereignisse protokolliert.

'F' (FAIL)
Es werden nur nicht erfolgreiche Ereignisse protokolliert.

'N' (NONE)
Es wird keine TAC-spezifische Art der SAT-Protokollierung definiert.

tacunit

enthält die Anzahl der Verrechnungseinheiten, die in der Abrechnungsphase des UTM-Accounting für jeden Aufruf dieses Transaktionscodes berechnet wird.
Die Verrechnungseinheiten werden auf den Verrechnungseinheitenzähler der Benutzerkennung aufaddiert, die den Transaktionscode aufgerufen hat.

tcbentry (nur auf BS2000-Systemen)

enthält den Namen der KDCDEF-Steueranweisung TCBENTRY, in der die TCB-Entries zusammengefasst sind, die diesem TAC zugeordnet sind.llected.

in_queue

ist nur bei Asynchron-TACs belegt.
gibt an, wieviele Asynchron-Nachrichten zur Zeit in der Message Queue des Transaktionscodes zwischengespeichert sind und vom zugehörigen Teilprogramm noch bearbeitet werden müssen.

Ist die Anzahl der Nachrichten größer als 99999, wird die Zahl nur unvollständig dargestellt. Deshalb sollte das Feld *in_queue_ex* verwendet werden (siehe in_queue_ex), weil hier auch größere Zahlen vollständig abgebildet werden.

used

gibt an, wieviele Teilprogrammläufe mit diesem Transaktionscode insgesamt bearbeitet wurden, seitdem der Zähler *used* das letzte Mal zurückgesetzt wurde.

Den Zähler können Sie mit KC_MODIFY_OBJECT auf 0 zurücksetzen. In UTM-S-Anwendungen wird *used* nur bei jeder Neugenerierung mit KDCDEF und bei jeder Änderungsgenerierung mit KDCDEF/KDCUPD automatisch auf 0 zurückgesetzt. In UTM-F-Anwendungen wird der Zähler *used* automatisch bei jedem Anwendungsstart auf 0 zurückgesetzt.

number_errors

gibt an, wieviele der Teilprogrammläufe, die über diesen Transaktionscode gestartet wurden, seit dem letzten Zurücksetzen des Zählers *number_errors* fehlerhaft beendet wurden.

Den Zähler können Sie mit KC_MODIFY_OBJECT auf 0 zurücksetzen.
In UTM-S-Anwendungen wird *number_errors* bei jeder Neugenerierung mit KDCDEF und bei jeder Änderungsgenerierung mit KDCDEF/KDCUPD auf 0 gesetzt. Bei UTM-F-Anwendungen wird *number_errors* bei jedem Anwendungsstart auf 0 gesetzt.

Ist die Anzahl der Teilprogrammläufe größer als 99999, wird die Zahl nur unvollständig dargestellt. Deshalb sollte das Feld *number_errors_ex* verwendet werden (siehe number_errors_ex), weil hier auch größere Zahlen vollständig abgebildet werden.

db_counter

enthält die mittlere Anzahl der Datenbankaufrufe aus Teilprogrammen, die seit dem letzten Zurücksetzen des Zählers *db_counter* über diesen Transaktionscode gestartet wurden.

db_counter ist bei Datenbankkopplung über die XA-Schnittstelle immer binär null.Den Zähler können Sie mit KC_MODIFY_OBJECT auf 0 zurücksetzen.

tac_elap_msec

gibt die mittlere Laufzeit der Teilprogramme an, die seit dem letzten Zurücksetzen des Zählers *tac_elap_msec* über diesen Transaktionscode gestartet wurden (elapsed time); Angabe in Millisekunden. Den Zähler können Sie mit KC_MODIFY_OBJECT auf 0 zurücksetzen.

db_elap_msec

enthält die mittlere Zeit, die für die Bearbeitung von Datenbankaufrufen in den Teilprogrammläufen mit diesem TAC benötigt wurde; Angabe in Millisekunden. *db_elap_msec* berücksichtigt alle Datenbankaufrufe seit dem letzten Zurücksetzen des Zählers.

db_elap_msec ist bei Datenbankkopplung über die XA-Schnittstelle immer binär null.
Den Zähler können Sie mit KC_MODIFY_OBJECT auf 0 zurücksetzen.

taccpu_msec

enthält die durchschnittliche CPU-Zeit in Millisekunden, die zur Bearbeitung dieses Transaktionscodes im Teilprogramm verbraucht wurde. Dies entspricht der von UTM verbrauchten CPU-Zeit plus der vom Datenbanksystem verbrauchten CPU-Zeit; Angabe in Millisekunden. *taccpu_msec* berücksichtigt alle Teilprogrammläufe seit dem letzten Zurücksetzen des Zählers.
Den Zähler können Sie mit KC_MODIFY_OBJECT auf 0 zurücksetzen.

deleted

gibt an, ob der Transaktionscode oder die TAC-Queue aus der Konfiguration gelöscht wurde oder nicht.

'Y' Der Transaktionscode oder die TAC-Queue wurde gelöscht. Der Name ist aber gesperrt. Es kann kein neuer Transaktionscode oder eine neue TAC-Queue mit diesem Namen erzeugt werden.

'N' Der Transaktionscode oder die TAC-Queue wurde nicht gelöscht.

pgwt

ist nur belegt, wenn in Ihrer Anwendung die Aufträge an TAC-Klassen prioritätengesteuert bearbeitet werden, d.h. die KDCDEF-Generierung enthält die Anweisung TAC-PRIORITIES.

pgwt gibt an, ob in einem Teilprogrammlauf, der für diesen Transaktionscode gestartet wird, blockierende Aufrufe (z.B. PGWT) durchgeführt werden dürfen.

'Y' Blockierende Aufrufe dürfen durchgeführt werden.

'N' Blockierende Aufrufe dürfen nicht durchgeführt werden.

encryption_level

ist nur für Vorgangs-TACs (*call_type*='F'oder 'B') relevant.
encryption_level gibt an, ob Nachrichten für diesen Transaktionscode verschlüsselt werden müssen oder nicht.

'N' (NONE)
Eine Nachrichtenverschlüsselung ist nicht erforderlich. Ein Client kann mit diesem Transaktionscode auch dann einen Service starten, wenn er die Eingabe-Nachricht unverschlüsselt überträgt. Die Ausgabe-Nachricht des Vorgangs wird nur verschlüsselt an den Client übertragen, wenn dessen Eingabe-Nachricht verschlüsselt war.

'2' (Level 2)
Für den Zugriff auf diesen Transaktionscode ist die Verschlüsselung der Eingabe-Nachrichten mit dem AES-CBC Algorithmus erforderlich.

'5' (Level 5) nur auf Unix-, Linux- und Windows-Systemen
Für den Zugriff auf diesen Transaktionscode ist die Verschlüsselung der Eingabenachrichten mit dem AES-GCM Algorithmus erforderlich.

Falls *encryption_level* = '2' oder '5' angegeben wird, dann kann ein Client über diesen Transaktionscode nur dann einen Vorgang (Service) starten, wenn er eine der folgenden Bedingungen erfüllt:

- Er ist als „trusted“ Client generiert (siehe *kc_pterm_str* oder *kc_tpool_str* Feld *encryption_level*). Ein „trusted“ Client kann über den Transaktionscode auch dann einen Vorgang starten, wenn er die Eingabe-Nachricht unverschlüsselt überträgt.
- Er hat die Eingabe-Nachricht mit mindestens der angegebenen Verschlüsselungsebene verschlüsselt an den Transaktionscode übertragen. Verschlüsselt ein „not-trusted" Client die erste Eingabe-Nachricht nicht oder nicht ausreichend oder unterstützt er keine Verschlüsselung, so wird kein Vorgang gestartet.

Alle Ausgabe-Nachrichten an einen not-trusted Client werden verschlüsselt an den Client übertragen. Wird der Transaktionscode durch Vorgangskettung gestartet, dann muss die erste Eingabe-Nachricht des Client nicht verschlüsselt sein.

Wird der Transaktionscode ohne Benutzerdaten aufgerufen oder durch Vorgangskettung gestartet, dann muss der Client die Fähigkeit zur Verschlüsselung haben. UTM verschlüsselt alle Dialog-Ausgabe-Nachrichten an den Client und erwartet, bei Mehrschritt-Vorgängen, alle weiteren Eingabe-Nachrichten von einem (not-trusted) Client verschlüsselt.

access_list

enthält den Namen eines Keysets, welches die Zugriffsrechte von Benutzern auf diesen Transaktionscode beschreibt.

Die Angabe von *access_list* bei TAC-Queues ist nicht erlaubt.

access_list und *lock_code* dürfen nicht gleichzeitig mit Werten belegt sein bzw. werden.

Ein Benutzer kann nur dann auf den Transaktionscode zugreifen, wenn das Keyset des Benutzers, das Keyset des LTERM-Partners, über den der Benutzer angemeldet ist, und das mit *access_list* bezeichnete Keyset mindestens einen gemeinsamen Keycode enthalten.

Sie können den Zugriffsschutz entfernen, indem Sie *access_list* mit Leerzeichen füllen. Enthält weder *access_list* noch *lock_code* einen Wert, kann jeder Benutzer auf den Transaktionscode zugreifen.

q_mode (**Q**ueue **Mode**)

bestimmt das Verhalten von UTM für den Fall, dass bereits die maximal erlaubte Anzahl von Nachrichten in einer Queue gespeichert und somit der Queue-Level erreicht ist.

'S' UTM lehnt weitere Aufträge ab.

'W' (nur bei *tac_type*='Q')

UTM nimmt weitere Nachrichten auf, löscht aber die ältesten in der Queue stehenden Nachrichten.

q_read_acl (only when *tac_type* ='Q')

zeigt die Rechte an (Name eines Keysets), die ein Benutzer benötigt, um Nachrichten aus dieser Queue lesen und löschen zu können.

Ein Benutzer kann nur dann lesend auf diese TAC-Queue zugreifen, wenn das Keyset des Benutzers und das Keyset des logischen Terminals, über das der Benutzer angemeldet ist, jeweils mindestens einen Keycode enthalten, der auch in dem angezeigten Keyset enthalten ist.

Enthält *q_read_acl* keinen Wert, dann können alle Benutzer Nachrichten aus dieser Queue lesen und dabei löschen.

q_write_acl (only when *tac_type*='Q')

zeigt die Rechte an (Name eines Keysets), die ein Benutzer benötigt, um Nachrichten in diese Queue zu schreiben.

Ein Benutzer kann nur dann schreibend auf diese TAC-Queue zugreifen, wenn das Keyset des Benutzers und das Keyset des logischen Terminals, über das der Benutzer angemeldet ist, jeweils mindestens einen Keycode enthalten, der auch in dem angezeigten Keyset enthalten ist.

Enthält *q_write_acl* keinen Wert, dann können alle Benutzer Nachrichten in diese Queue schreiben.

nbr_dputs

Anzahl anstehender zeitgesteuerter Aufträge für diesen TAC, deren Startzeitpunkt noch nicht erreicht ist.

nbr_ack_jobs

Anzahl anstehender Quittierungsaufträge für diesen TAC, die noch nicht aktiviert wurden.

dead_letter_q

gibt an, ob eine Asynchron-Nachricht in der Dead Letter Queue aufbewahrt werden soll, wenn sie nicht ordnungsgemäß verarbeitet wurde und keine Redelivery erfolgt ist.

'Y' Fehlerhafte Asynchron-Nachrichten werden in der Dead Letter Queue gesichert.

dead_letter_q = '*Y*' ist nicht erlaubt für KDCDLETQ, KDCMSGTC, alle Dialog-TACs und Asynchron-TACs mit CALL=NEXT.

'N' Fehlerhafte Asynchron-Nachrichten werden gelöscht, wenn keine Redelivery erfolgt ist.

nbr_ta_commits

Anzahl der Teilprogrammläufe zu diesem TAC, die eine Transaktion erfolgreich abgeschlossen haben.

Den Zähler können Sie mit KC_MODIFY_OBJECT auf 0 zurücksetzen.

number_errors_ex

siehe number_errors.

in_queue_ex

siehe in_queue.

taccpu_micro_sec

enthält die durchschnittliche CPU-Zeit in Mikrosekunden, die zur Bearbeitung dieses Transaktionscodes im Teilprogramm verbraucht wurde. Dies entspricht der von UTM verbrauchten CPU-Zeit plus der vom Datenbanksystem verbrauchten CPU-Zeit.
taccpu_micro_sec berücksichtigt alle Teilprogrammläufe seit dem letzten Zurücksetzen des Zählers.
Den Zähler können Sie mit KC_MODIFY_OBJECT auf 0 zurücksetzen.

11.3.1.30 kc_tacclass_str - TAC-Klassen der Anwendung

Für den Objekttyp KC_TACCLASS ist die Datenstruktur *kc_tacclass_str* definiert. In *kc_tacclass_str* liefert UTM bei KC_GET_OBJECT die folgenden Informationen zurück:

- Eigenschaften der TAC-Klasse.
- Statistikinformationen darüber, wie oft und wie lange Aufträge für die TAC-Klasse auf die Bearbeitung warten mussten.
- wenn die Anwendung ohne Prioritätensteuerung generiert ist (d.h. ohne Anweisung TAC-PRIORITIES), die aktuelle Anzahl der Prozesse, die maximal gleichzeitig Aufträge für die Transaktionscodes der TAC-Klasse bearbeiten dürfen.

mod[1]	**Datenstruktur kc_tacclass_str**
-	`char tacclass[2];`
x(A)[2]	`char tasks[3];`
x(A)[2]	`char tasks_free[3];`
-	`char pgwt;`
-	`char waiting_msgs[10];`
x(GIR)	`char avg_wait_time_msec[10];`
-	`char prio[3];`
x(GIR)	`nr_calls[10];`
x(GIR)	`nr_waits[10];`

[1] Feldinhalt mit KC_MODIFY_OBJECT modifizierbar; siehe Abschnitt "obj_type=KC_TACCLASS".

[2] Diese Eigenschaften sind nur modifizierbar, wenn die Anwendung ohne TAC-PRIORITIES-Anweisung generiert ist. Es darf in einem KC_MODIFY_OBJECT-Aufruf nur eines dieser Felder belegt werden.

Die Felder der Datenstruktur haben die folgende Bedeutung:

tacclass

enthält die Nummer der TAC-Klasse. Für *tacclass* wird eine Zahl zwischen 1 und 16 ausgegeben.

Die TAC-Klassen 1 - 8 sind Dialog-TAC-Klassen.
Die TAC-Klassen 9 - 16 sind Asynchron-TAC-Klassen.

tasks ist nur relevant, wenn für die TAC-Klassen keine Prioritätensteuerung generiert ist (KDCDEF-Generierung ohne TAC-PRIORITIES-Anweisung).

tasks gibt an, wieviele Prozesse der Anwendung gleichzeitig TACs der TAC-Klasse *tacclass* bearbeiten dürfen (absolute Angabe).

Siehe auch "obj_type=KC_TACCLASS".

Ist die Anwendung mit Prioritätensteuerung generiert, dann enthält *tasks* ein Leerzeichen.

tasks_free

ist nur relevant, wenn die Anwendung ohne TAC-PRIORITIES-Anweisung generiert ist.

tasks_free enthält bei Dialog-TAC-Klassen die Anzahl der Prozesse der Anwendung, die mindestens für die Verarbeitung von Transaktionscodes anderer TAC-Klassen freigehalten werden sollen. Bei Asynchron-TAC-Klassen enthält *tasks_free* die Anzahl der Prozesse, die mindestens für die Bearbeitung von Transaktionscodes anderer Asynchron-TAC-Klassen freigehalten werden sollen.

UTM liefert in *tasks_free* '0' zurück, wenn der Wert von *tasks_free* weder bei der KDCDEF-Generierung noch durch die Administration festgelegt wurde oder wenn bei der letzten Änderung der Prozesszahl für die TAC-Klasse ein Wert für *tasks* festgelegt wurde.

Siehe auch "obj_type=KC_TACCLASS".

Ist die Anwendung mit Prioritätensteuerung generiert, dann enthält *tasks_free* Leerzeichen.

pgwt gibt an, ob in dieser TAC-Klasse Teilprogramme ablaufen dürfen, die blockierende Aufrufe wie z.B. den KDCS-Aufruf PGWT enthalten.

'Y' In dieser TAC-Klasse sind blockierende Aufrufe zugelassen.

'N' In dieser TAC-Klasse sind blockierende Aufrufe nicht zugelassen.

Teilprogramme, die blockierende Aufrufe enthalten, sind höchstens in einer Dialog-TAC-Klasse und in einer Asynchron-TAC-Klasse erlaubt.

waiting_msgs

enthält die Anzahl der Aufträge, die derzeit für Transaktionscodes dieser TAC-Klasse in UTM zwischengespeichert und noch nicht bearbeitet sind.

avg_wait_time_msec

enthält die mittlere Wartezeit der Aufträge in den Auftragswarteschlangen, die den Transaktionscodes dieser TAC-Klasse zugeordnet sind.

Wenn kein Prozess für die TAC-Klasse zur Verfügung steht, nimmt UTM Aufträge für die TAC-Klasse entgegen (mit freien Prozessen, die keine Aufträge an diese TAC-Klasse bearbeiten „dürfen“) und speichert sie in der KDCFILE zwischen. Das ist immer dann der Fall, wenn Aufträge für TAC-Klassen höherer Priorität anstehen (bei Prioritätensteuerung) bzw. (bei Prozessbegrenzung) bereits die maximal erlaubte Anzahl an Prozessen Transaktionscodes der TAC-Klasse bearbeiten (siehe *tasks*, *tasks_free*).

Die Zeit zwischen Entgegennehmen eines Auftrags und dem Start seiner Bearbeitung ist die hier angezeigte Wartezeit.

Die Einheit des Werts *avg_wait_time_msec* ist Millisekunde.

Der Wert von *avg_wait_time_msec* kann auf 0 zurückgesetzt werden. Bei einem Zurücksetzen dieses Werts wird implizit auch die Werte *nr_calls* und *nr_waits* zurückgesetzt.

prio enthält die Art der Prioritätensteuerung, die für die TAC-Klassen generiert wurde.
Folgende Werte sind möglich:

'ABS' Absolute Prioritäten:
Ein freier Prozess wird immer der TAC-Klasse mit der höchsten Priorität, also 1 bzw. 9 zugeordnet, sofern es dort wartende Aufträge gibt. Die TAC-Klasse mit der nächst niedrigeren Priorität wird erst bedient, wenn es in der TAC-Klasse mit der höchsten Priorität keine wartenden Aufträge mehr gibt.

'REL' Relative Prioritäten:
Freie Prozesse werden öfter TAC-Klassen hoher als TAC-Klassen niedriger Priorität zugeordnet, sofern für diese wartende Aufträge vorhanden sind.

'EQ' Gleiche Prioritäten:
Sofern Aufträge vorhanden sind, werden alle TAC-Klassen gleich häufig bedient.

'NO' Es ist keine Prioritätensteuerung generiert.

nr_calls

Anzahl von Teilprogrammläufen für diese TAC-Klasse.

Sie können den Wert mit KC_MODIFY_OBJECT auf 0 zurücksetzen. Bei einem Zurücksetzen dieses Werts werden implizit auch die Werte *avg_wait_time_msec* und *nr_waits* zurückgesetzt.

nr_waits

Anzahl von Wartesituationen, die für die Berechnung des Wertes *avg_wait_time_msec* berücksichtigt wurden.

Sie können den Wert mit KC_MODIFY_OBJECT auf 0 zurücksetzen. Bei einem Zurücksetzen dieses Werts werden implizit auch die Werte *avg_wait_time_msec* und *nr_calls* zurückgesetzt.

11.3.1.31 kc_tpool_str - LTERM-Pools der Anwendung

Für den Objekttyp KC_TPOOL ist die Datenstruktur *kc_tpool_str* definiert. In *kc_tpool_str* liefert UTM bei KC_GET_OBJECT die folgenden Informationen über einen LTERM-Pool zurück:

- Anzahl der LTERM-Partner, die zur Zeit für den LTERM-Pool zugelassen sind.
- Eigenschaften der LTERM-Partner des LTERM-Pools.
- Typ der Clients, die sich über diesen LTERM-Pool an die Anwendung anschließen können.
- Statistikdaten über die Auslastung des LTERM-Pools.

mod[1]	**Datenstruktur kc_tpool_str**
-	`char lterm[8];`
-	`char pronam[8];`
-	`char ptype[8];`
-	`char bcamappl[8];`
-	`char connect_mode;`
-	`char max_number[10];`
-	`char kset[8];`
-	`char locale_lang_id[2];` (nur auf BS2000-Systemen)
-	`char locale_terr_id[2];` (nur auf BS2000-Systemen)
-	`char locale_ccsname[8];` (nur auf BS2000-Systemen)
-	`char lock_code[4];`
x(GP)[2]	`char state;`
x(GP)[2]	`char state_number[10];`
-	`char format_attr;` (nur auf BS2000-Systemen)
-	`char format_name[7];` (nur auf BS2000-Systemen)
-	`char qlev[5];`
-	`char termn[2];`
-	`char annoamsg;` (nur auf BS2000-Systemen)
-	`char netprio;` (nur auf BS2000-Systemen)
-	`char protocol;` (nur auf BS2000-Systemen)

mod[1]	Datenstruktur kc_tpool_str
-	`char actcon[10];`
-	`char maxcon[10];`
-	`char map;`
x(GP)	`char idletime[5];`
-	`char encryption_level;`
-	`char user_kset[8];`
-	`char usp_hdr;`
-	`char kerberos_dialog;` (nur auf BS2000-Systemen)
-	`char pronam_long[64];`
-	`char resolve_names;`

[1] Feldinhalt mit KC_MODIFY_OBJECT modifizierbar; siehe Abschnitt "obj_type=KC_TPOOL".

[2] Bei KC_MODIFY_OBJECT müssen beide Felder zusammen angegeben werden.

Die Felder der Datenstruktur haben die folgende Bedeutung:

lterm

enthält das Präfix für die Namen der LTERM-Partner des LTERM-Pools. Die Namen der LTERM-Partner setzen sich zusammen aus diesem Präfix und einer Laufnummer. Die Laufnummer geht von 1 bis zu dem Wert, der in *max_number* zurückgeliefert wird.

Beispiel:
Ist *max_number*='1000' und *lterm*='LTRM', dann heißen die LTERM-Partner des LTERM-Pools LTRM0001, LTRM0002, ..., LTRM1000.

pronam

gibt an, auf welchem Rechner sich die Clients befinden müssen, um sich über diesen LTERM-Pool an die Anwendung anschließen zu können.

Wenn für den LTERM-Pool ein Rechnername mit mehr als 8 Zeichen generiert wurde, dann kann der vollständige, bis zu 64 Zeichen lange Name dem Feld *pronam_long* entnommen werden. Das Feld *pronam* enthält in diesem Fall die ersten 8 Zeichen dieses Namens.

UTM liefert entweder den symbolischen Namen zurück, unter dem der Rechner dem lokalen Transportsystem bekannt ist, oder bei einem offenen LTERM-Pool den Wert '*ANY'. '*ANY' bedeutet:

Über den LTERM-Pool kann sich jeder Client an die Anwendung anmelden, der die folgenden Bedingungen erfüllt:

- Sein Terminaltyp stimmt mit der Angabe in *ptype* überein.

- Er wurde nicht explizit in die Konfiguration eingetragen (mit der KDCDEF-Anweisung PTERM oder dynamisch mit Objekttyp KC_PTERM).
- Für den Rechner, auf dem der Client residiert, und seinen Terminaltyp (*ptype*) existiert kein anderer LTERM-Pool.

ptype

Typ der Clients, die sich über diesen LTERM-Pool an die Anwendung anschließen dürfen. Die Bedeutung der Werte, die UTM in *ptype* zurückliefert, entnehmen Sie bitte den Tabellen für BS2000-Systeme und für Unix-, Linux- und "Windows-Systeme im Abschnitt "kc_pterm_str - Clients und Drucker".

Nur auf BS2000-Systemen:

Ist *ptype*='*ANY', dann handelt es sich um einen offenen LTERM-Pool. An den LTERM-Pool können sich alle Clients anschließen, die sich auf bzw. an dem Rechner *pronam* befinden und für die Folgendes gilt:

- Der Client wurde nicht explizit in die Konfiguration eingetragen.
- Für den Rechner in *pronam* existiert kein LTERM-Pool, für den in *ptype* der Typ des Client gesetzt ist.

bcamappl

Name der lokalen UTM-Anwendung (BCAMAPPL-Name), über den die Verbindung zwischen Client und UTM-Anwendung aufgebaut wird.

Diesen Namen muss der Client angeben, wenn er eine Verbindung zur lokalen Anwendung aufbauen will.

connect_mode

legt fest, ob sich ein Client mehrfach unter demselben Namen über den LTERM-Pool an die UTM-Anwendung anschließen kann.

'S' Jeder Client kann sich nur einmal unter demselben Namen an den LTERM-Pool anschließen.

'M' Ein UPIC-Client (*ptype*='UPIC-R' oder 'UPIC-L') bzw. eine TS-Anwendung (='APPLI' oder 'SOCKET'), der/die mehrfach auf demselben Rechner abläuft, kann sich mehrfach unter demselben Namen über den LTERM-Pool an die UTM-Anwendung anschließen. Es muss nicht für jede Verbindung ein neuer Name erzeugt werden.

Der UPIC-Client bzw. die TS-Anwendung kann sich maximal so oft an den LTERM-Pool anschließen wie LTERM-Partner für den LTERM-Pool zugelassen sind. Der Name des jeweiligen Pool-LTERM-Partners wird in diesem Fall beim Verbindungsaufbau mit dem Namen des Client/der TS-Anwendung gleichgesetzt, d.h. die Anwendung identifiziert den Partner über das Tripel (Name des LTERM-Partners, *pronam* und *bcamappl)*. Der UPIC-Client bzw. die TS-Anwendung ist nicht unter seinem lokalen Namen oder seinem Anwendungsnamen in der UTM-Anwendung bekannt.

max_number

gibt an, wieviele Clients sich maximal gleichzeitig über diesen LTERM-Pool anschließen dürfen. D.h. *max_number* gibt an, wieviele LTERM-Partner in diesem LTERM-Pool zusammengefasst sind.

kset enthält den Namen des Keysets, das dem LTERM-Pool zugeordnet ist. Das Keyset legt fest, welche Transaktionscodes die Clients aufrufen dürfen, die sich über diesen LTERM-Pool an die Anwendung anschließen. Die Clients dürfen einen Transaktionscode nur starten, wenn das Keyset einen Keycode

enthält, der im Zahlenwert mit dem Lockcode des Transaktionscodes übereinstimmt, oder der Transaktionscode nicht zugriffsgesichert ist, d.h. keinen Lockcode besitzt.

Ist dem LTERM-Pool kein Keyset zugeordnet, dann ist *kset* mit Leerzeichen belegt.

Bei *ptype*='UPIC-...', 'APPLI' oder 'SOCKET' gilt Folgendes:

kset legt die maximalen Zugriffsrechte eines Clients fest, der sich über diesen LTERM-Pool anschließt.

kset wird immer wirksam, wenn der Client beim Session-/Conversationaufbau eine echte Benutzkennung an UTM übergibt. Die Zugriffsrechte ergeben sich dann aus der Menge der Keycodes, die sowohl in dem Keyset der Benutzerkennung als auch in *kset* enthalten sind.

Übergibt der Client für die Session/Conversation keine echte Benutzerkennung an openUTM, dann ergeben sich seine Zugriffsrechte aus der Schnittmenge der Keycodes in *kset* und *user_kset* (minimale Zugriffsrechte).

locale_lang_id, locale_terr_id, locale_ccsname (nur auf BS2000-Systemen)

enthalten die drei Komponenten des Locale, das dem LTERM-Pool zugeordnet ist. Das Locale definiert die Sprachumgebung der Clients, die sich über diesen LTERM-Pool an die Anwendung anschließen (siehe auch openUTM-Handbuch „Anwendungen generieren“).

locale_lang_id
enthält das bis zu 2 Zeichen lange Sprachkennzeichen.

locale_terr_id
enthält ein bis zu 2 Zeichen langes Territorialkennzeichen.

locale_ccsname
(**c**oded **c**haracter **s**et **name**)
enthält den bis zu 8 Zeichen langen Namen eines erweiterten Zeichensatzes (CCS-Name; siehe auch Benutzerhandbuch zu XHCS).

lock_code

enthält den Lockcode, der den LTERM-Partnern des LTERM-Pools zugeordnet ist (Zugriffsschutz). Nur Benutzer/Clients, die den entsprechenden Keycode besitzen, dürfen sich über diesen LTERM-Pool anschließen.
lock_code kann eine Zahl zwischen '0' und '4000' enthalten. '0' bedeutet, dass der LTERM-Pool nicht durch einen Lockcode geschützt ist.

state, state_number

Bei der KDCDEF-Generierung des LTERM-Pools wird die Anzahl der LTERM-Partner festgelegt, die in diesem LTERM-Pool zusammengefasst sind (siehe max_number). Die Anzahl der LTERM-Partner, über die sich Clients an die Anwendung anschließen können, kann jedoch im Betrieb durch die Administration auf einen kleineren Wert gesetzt werden. Der Rest der LTERM-Partner wird dadurch gesperrt. In den Feldern *state* und *state_number* gibt UTM an, wieviele LTERM-Partner des LTERM-Pools zur Zeit zugelassen, d.h. nicht gesperrt sind. Die Anzahl der zugelassenen LTERM-Partner bestimmt, wieviele Clients sich gleichzeitig über diesen LTERM-Pool an die Anwendung binden können.

Enthält *state* den Wert 'Y', wird der Pool für die Anzahl an Kommunikationspartnern zugelassen, die in *state_number* angegeben ist (ON). Enthält *state* den Wert 'N', wird der Pool für die Anzahl an Kommunikationspartnern gesperrt, die in *state_number* angegeben ist (OFF).

Sind alle LTERM-Partner des LTERM-Pools gesperrt, dann enthält *state* den Wert 'Y' und *state_number* den Wert '0'.

format_attr, format_name (nur auf BS2000-Systemen)

definieren das Startformat für Benutzer an Terminals, die sich über den LTERM-Pool anschließen. Nach dem Aufbau der Verbindung zwischen Terminal und Anwendung wird das in *format_attr* und *format_name* beschriebene Format am Terminal ausgegeben, sofern kein Terminal-spezifischer Wiederanlauf durchgeführt wird.

format_attr
enthält das Formatkennzeichen:

'A' (Formatattribut ATTR)
Das Startformat ist ein Format mit Benutzerattributen. Die Eigenschaften der Formatfelder können vom KDCS-Teilprogramm verändert werden. Der Formatname an der Programmschnittstelle KDCS ist +*formatname*.

'N' (Formatattribut NOATTR)
Das Startformat ist ein Format ohne Benutzerattribute. Weder Feld- noch Formateigenschaften können von KDCS-Teilprogrammen verändert werden. Der Formatname an der Programmschnittstelle KDCS ist **formatname*.

'E' (Formatattribut EXTEND)
Das Startformat ist ein Format mit erweiterten Benutzerattributen. Es können sowohl die Eigenschaften der Formatfelder als auch globale Formateigenschaften von KDCS-Teilprogrammen verändert werden. Der Formatname an der Programmschnittstelle KDCS ist #*formatname*.

format_name
enthält den Namen des Startformats. Der Name kann bis zu 7 Zeichen lang sein und enthält nur alphanumerische Zeichen.

qlev (**q**ueue **lev**el)

gibt an, wieviele Asynchron-Nachrichten UTM maximal gleichzeitig in der Message Queue eines LTERM-Partners des Pools zur Bearbeitung zwischenspeichert. Wird der Schwellwert für einen LTERM-Partner des LTERM-Pools überschritten, dann weist UTM weitere Asynchron-Aufträge an diesen LTERM-Partner ab. Der Schwellwert wird bei der KDCDEF-Generierung festgelegt.

termn (**ter**minal **mn**emonic)

enthält das Kennzeichen für die Art der Clients, die sich über diesen LTERM-Pool anschließen können. Das Kennzeichen stellt UTM KDCS-Teilprogrammen beim Ablauf im Feld KCTERMN des KB-Kopfes zur Verfügung, die über den LTERM-Pool gestartet werden. Das Kennzeichen ist maximal 2 Zeichen lang. Die Standardwerte für *termn* können Sie der Tabelle bei *ptype* im Abschnitt "kc_pterm_str - Clients und Drucker" ("BS2000-Systeme, Unix-, Linux-" und "Windows-Systeme") entnehmen.

annoamsg (**anno**unce **a**synchronous **m**e**s**sa**g**e, nur auf BS2000-Systemen))

gibt an, ob UTM Asynchron-Nachrichten vor der Ausgabe am Terminal durch eine Meldung in der Systemzeile ankündigt.

'Y'

UTM kündigt jede Asynchron-Nachricht an das Terminal mit der Meldung K012 in der Systemzeile an. Der Benutzer muss die Asynchron-Nachricht dann explizit mit dem Kommando KDCOUT anfordern.

'N' Asynchron-Nachrichten an das Terminal werden sofort, ohne Ankündigung gesendet. Bei *annoamsg* ='N' ist der Verbindungsaufbau zu diesem LTERM-Pool über eine Multiplexverbindung erst ab OMNIS V7.0 möglich.

netprio (nur auf BS2000-Systemen)

gibt die Transportpriorität an, die auf Transportverbindungen zwischen der Anwendung und den Clients benutzt wird, die sich über diesen LTERM-Pool anschließen.

'M' Transportpriorität „Medium“

'L' Transportpriorität „Low“

protocol (nur auf BS2000-Systemen)

gibt an, ob auf Verbindungen zwischen der UTM-Anwendung und einem Client, der sich über diesen LTERM-Pool anschließt, mit dem Benutzerdienstprotokoll NEABT gearbeitet wird.

'N' Zwischen der UTM-Anwendung und dem Client wird ohne Benutzerdienstprotokoll gearbeitet.

Für UPIC-Clients (*ptype*='UPIC-R') und TS-Anwendungen (*ptype*='APPLI' oder 'SOCKET') wird immer *protocol*='N' ausgegeben.
Zu einem LTERM-Pool mit *protocol*='N' können keine Verbindungen über einen Multiplexanschluss aufgebaut werden.

'S' (STATION)
Zwischen der UTM-Anwendung und dem Client wird mit dem Benutzerdienstprotokoll (NEABT) gearbeitet.

UTM verwendet das Benutzerdienstprotokoll NEABT bei LTERM-Pools mit *ptype*='*ANY' z.B. zur Bestimmung des Typs (*ptype*) eines Clients. In diesem Fall wird immer mit NEABT gearbeitet.

actcon

gibt an, wieviele Clients zur Zeit über diesen LTERM-Pool mit der Anwendung verbunden sind.

maxcon

enthält die maximale Anzahl der Clients, die im aktuellen Anwendungslauf gleichzeitig über diesen LTERM-Pool mit der Anwendung verbunden waren.

Der Zähler wird bei jedem Start der Anwendung auf 0 zurückgesetzt.

map

gibt an, ob UTM für Benutzer-Nachrichten, die zwischen den Partner-Anwendungen ausgetauscht werden, eine Code-Konvertierung (ASCII <-> EBCDIC) durchführt.
Benutzer-Nachrichten werden an der KDCS-Schnittstelle bei den Aufrufen zur Nachrichtenbehandlung (MPUT/FPUT/DPUT) im Nachrichtenbereich übergeben.

'U'

(USER)
UTM konvertiert die Benutzer-Nachrichten nicht, d.h. die Nachrichten werden unverändert zwischen den Partner-Anwendungen übertragen.

'1', '2', '3', '4' (SYS1 | SYS2 | SYS3 | SYS4)

ist nur für folgende TS-Anwendungen erlaubt:

- BS2000-Systeme: *ptype*='SOCKET'
- Unix-, Linux- oder Windows-Systeme: *ptype*='APPLI' oder 'SOCKET'

UTM konvertiert die Benutzernachrichten gemäß den für die Code-Konvertierung bereitgestellten Konvertierungstabellen, siehe Abschnitt „Code-Konvertierung“ im openUTM-Handbuch „Anwendungen generieren“, d.h.:

- Vor dem Senden wird auf Unix-, Linux- und Windows-Systemen von ASCII nach EBCDIC und auf BS2000-Systemen von EBCDIC nach ASCII konvertiert.
- Nach dem Empfangen wird auf Unix-, Linux- und Windows-Systemen von EBCDIC nach ASCII und auf BS2000-Systemen von ASCII nach EBCDIC konvertiert.

Dabei geht UTM davon aus, dass die Nachricht nur abdruckbare Zeichen enthält.
Weitere Informationen zur Code-Konvertierung finden Sie im openUTM-Handbuch „Anwendungen programmieren mit KDCS“, Abschnitt "Code-Konvertierung".

idletime

Zeit in Sekunden, die UTM nach dem Ende einer Transaktion oder nach dem Anmelden (KDCSIGN) maximal auf eine Eingabe vom Client wartet. Bei Zeitüberschreitung wird die Verbindung zum Client abgebaut. Ist der Client ein Terminal, dann wird vor dem Verbindungsabbau die Meldung K021 ausgegeben.

Der Wert 0 bedeutet Warten ohne Zeitbegrenzung.

encryption_level

nur relevant für UPIC-Clients und auf BS2000-Systemen für einige Terminalemulationen.

encryption_level gibt an, ob die UTM-Anwendung auf der Verbindung zum Client, der sich über den LTERM-Pool an die Anwendung anschließen will,

- die Verschlüsselung der Nachrichten standardmäßig anfordert oder nicht,
- welche Verschlüsselungsebene dabei gefordert wird,
- oder ob die Clients „trusted“ Clients sind.

Folgende Werte sind möglich:

'N' (NONE)
UTM fordert **keine** Verschlüsselung der Nachrichten an.
Services, für die die Verschlüsselung generiert wurde (siehe *kc_tac_str.encryption_level* im Abschnitt "kc_tac_str - Transaktionscodes lokaler Services"), können von einem Client, der sich über diesen Pool anschließt, nur gestartet werden, wenn der Client beim Verbindungsaufbau die Verschlüsselung auswählt.
Passworte werden verschlüsselt übertragen, sofern beide Partner Verschlüsselung unterstützen.

'3' (LEVEL 3)
UTM fordert von jedem Client die Verschlüsselung der Nachrichten mit der Verschlüsselungsebene 3 an, d.h. die Nachrichten werden mit dem AES-CBC Algorithmus verschlüsselt und zum Austausch des AES-Schlüssels wird ein RSA-Schlüssel mit einer Schlüssellänge von 1024 bit verwendet.
Der Verbindungsaufbau zu dem Client wird von UTM abgelehnt, wenn der Client nicht mindestens diese Verschlüsselungsebene unterstützt.

'4' (LEVEL 4)
UTM fordert von jedem Client die Verschlüsselung der Nachrichten mit der Verschlüsselungsebene 4 an, d.h. die Nachrichten werden mit dem AES-CBC Algorithmus verschlüsselt und zum Austausch des AES-Schlüssels wird ein RSA-Schlüssel mit einer Schlüssellänge von 2048 bit verwendet.
Der Verbindungsaufbau zu dem Client wird von UTM abgelehnt, wenn der Client nicht mindestens diese Verschlüsselungsebene unterstützt.

'5' (LEVEL 5) nur auf Unix-, Linux- und Windows-Systemen
Schlüssellänge wie bei LEVEL 4. Zur Vereinbarung des Session-Keys wird das auf Elliptic Curves basierende Diffie-Hellman Verfahren verwendet und Ein-/Ausgabe-Nachrichten werden mit dem AES-GCM Algorithmus verschlüsselt. Der Verbindungsaufbau zu dem Client wird von UTM abgelehnt, wenn der Client nicht mindestens diese Verschlüsselungsebene unterstützt

'T' (TRUSTED)
Die Clients sind „trusted Clients“. Nachrichten und Passworte zwischen einem Client und einer Anwendung werden nicht verschlüsselt.

Ein „trusted Client“ kann auch Vorgänge starten, deren Vorgangs-TACs Verschlüsselung erfordern (generiert mit *kc_tac_str.encryption_level* = '2' oder '5'; siehe Abschnitt "kc_tac_str - Transaktionscodes lokaler Services").
Verbindungen, die über einen Transportsystemendpunkt (BCAMAPPL) aufgebaut werden, der mit T-PROT=(..., SECURE) generiert ist, werden von UTM immer als trusted eingestuft.

user_kset

nur relevant für *ptype*='UPIC-...', 'APPLI' oder 'SOCKET'.

user_kset enthält den Namen des Keyset, das die minimalen Zugriffsrechte des Client in der lokalen Anwendung festlegt.

Das in *user_kset* angegebene Keyset wird dann wirksam, wenn der Client unter der Verbindungs-Benutzerkennung angemeldet ist (siehe auch *kset*).

Es gelten immer die Zugriffsrechte in *kset*.

usp_hdr

zeigt an, für welche Ausgabe-Nachrichten UTM auf dieser Verbindung einen UTM-Socket-Protokoll-Header aufbaut. Mögliche Werte sind:

'A' (ALL)
Bei allen Ausgabe-Nachrichten (Dialog, Asynchron, K-Meldungen) erzeugt UTM einen UTM-Socket-Protokoll-Header und stellt diesen der Nachricht voran.

'M'

(MSG)
Nur bei Ausgabe von K-Meldungen erzeugt UTM einen UTM-Socket-Protokoll-Header und stellt diesen der Nachricht voran.

'N' (NO)
UTM erzeugt für keine Ausgabe-Nachricht einen UTM-Socket-Protokoll-Header.

Die Werte 'A' und 'M' können nur bei LTERM-Pools vorkommen, die für die Kommunikation über Socket-Verbindungen konfiguriert sind (*ptype*='SOCKET').

kerberos_dialog (nur auf BS2000-Systemen)

'Y' Beim Verbindungsaufbau wird für Clients, die Kerberos unterstützen und die sich direkt (nicht über OMNIS) über diesen Terminal-Pool an die Anwendung anschließen, ein Kerberos-Dialog durchgeführt.

'N' Es wird kein Kerberos-Dialog durchgeführt.

Detaillierte Information dazu entnehmen Sie dem openUTM-Handbuch „Anwendungen generieren“.

pronam_long

gibt an, auf welchem Rechner sich die Clients befinden müssen, um sich über diesen LTERM-Pool an die Anwendung anschließen zu können.

UTM liefert entweder den symbolischen Namen zurück, unter dem der Rechner dem lokalen Transportsystem bekannt ist, oder bei einem offenen LTERM-Pool den Wert '*ANY'.

'*ANY' bedeutet, dass sich jeder Client über den LTERM-Pool an die Anwendung anmelden kann, der die folgenden Bedingungen erfüllt:

- Sein Typ stimmt mit der Angabe in *ptype* überein.
- Er wurde nicht explizit in die Konfiguration eingetragen (mit der KDCDEF-Anweisung PTERM oder dynamisch mit dem Objekttyp KC_PTERM).
- Für den Rechner, auf dem der Client residiert, und seinen Terminaltyp (*ptype*) existiert kein anderer LTERM-Pool.

resolve_names

gibt an, ob nach einem Verbindungsaufbau eine Namensauflösung per DNS stattfinden soll oder nicht. Siehe KDCDEF-Anweisung SUBNET.

11.3.1.32 kc_transfer_syntax_str - Transfersyntax für die Kommunikation über OSI TP

Für den Objekttyp KC_TRANSFER_SYNTAX ist die Datenstruktur *kc_transfer_syntax_str* definiert. In *kc_transfer_syntax_str* liefert UTM bei KC_GET_OBJECT den lokalen Namen und den Object Identifier einer Transfersyntax zurück.
Die Transfersyntax gibt bei der Kommunikation über OSI TP an, in welcher Form die Benutzerdaten zum Kommunikationspartner übertragen werden. Beide Kommunikationspartner müssen auf einer Verbindung dieselbe Transfersyntax verwenden.

Datenstruktur kc_transfer_syntax_str
`char transfer_syntax_name[8];`
`char object_id[10][8];`

Die Felder der Datenstruktur haben die folgende Bedeutung:

transfer_syntax_name

enthält den Namen, der lokal für die Transfersyntax generiert wurde. Er ist maximal 8 Zeichen lang.

object_id

enthält den Object-Identifier der Transfersyntax.

Der Object-Identifier besteht aus mindestens 2, maximal aber aus 10 Komponenten. Die einzelnen Komponenten sind positive ganze Zahlen im Bereich von 0 bis 67108863.

UTM liefert pro Komponente des Object-Identifiers ein Feldelement zurück, d.h. die Anzahl der belegten Feldelemente in *object_id* entspricht der Anzahl der Komponenten. Die restlichen Feldelemente sind mit binär null versorgt.

Näheres zum Object-Identifier finden Sie im openUTM-Handbuch „Anwendungen generieren“.

11.3.1.33 kc_user_str, kc_user_fix_str, kc_user_dyn1_str bzw. kc_user_dyn2_str - Benutzerkennungen

Für die Objekttypen KC_USER, KC_USER_FIX, KC_USER_DYN1 und KC_USER_DYN2 sind die Datenstrukturen *kc_user_str*, *kc_user_fix_str*, *kc_user_dyn1_str* und *kc_user_dyn2_str* definiert. Die Datenstruktur *kc_user_str* ist in drei Teilstrukturen aufgeteilt, um die Performance beim Zugriff auf die User-Daten in UTM-Cluster-Anwendungen zu verbessern. Alle Daten, die bei UTM-Cluster-Anwendungen in der Cluster-User-Datei gehalten werden, liegen in die Datenstruktur *kc_user_dyn2_str*.

Benutzerkennungen können dynamisch mit KC_CREATE_OBJECT erzeugt, mit KC_DELETE_OBJECT gelöscht oder mit KC_MODIFY_OBJECT modifiziert werden.

Zum Erzeugen und Modifizieren müssen Sie die Struktur *kc_user_str* verwenden. Die anderen Strukturen sind nur zum Lesen mit KC_GET_OBJECT vorgesehen.

In *kc_user_str, kc_user_fix_str, kc_user_dyn1_str* bzw. *kc_user_dyn2_str* liefert bei KC_GET_-OBJECT UTM folgende Informationen über eine Benutzerkennung zurück:

- Attribute dieser Benutzerkennung, wie z.B. Art und Weise der Authentisierung (Passwort, Magnetstreifenkarte), Startformat, Zugriffsrechte, Administrationsberechtigung.
- Anzahl der Aufträge, die über diese Benutzerkennung eingegeben wurden, und Statistikdaten über den Ressourcenbedarf bei der Bearbeitung der Aufträge.
- Anzahl der Asynchron-Aufträge, die unter der Benutzerkennung ablaufen.
- Anzahl der Benutzer, die zur Zeit unter dieser Benutzerkennung bei der Anwendung angemeldet sind, sowie der Zeitpunkt der letzten Anmeldung unter dieser Benutzerkennung.
- Anzahl der Sicherheitsverletzungen von Benutzern/Clients, die sich über diese Benutzerkennung angemeldet haben.
- Eigenschaften der zugehörigen USER-Queue.

mod [1]	**Datenstruktur kc_user_str**	**siehe** [2]
-	`char us_name[8];`	us_name
x(GPD)	`char kset[8];`	kset
x(GPD)	`char state;`	state
-	`char card_position[3];` (nur auf BS2000-Systemen)	card_position
-	`char card_string_lth[3];` (nur auf BS2000-Systemen)	card_string_lth
-	`char card_string_type;` (nur auf BS2000-Systemen)	card_string_type
-	`union kc_string card_string;`	card_string
x(GPD)	`union kc_pw password;`	password
x(GPD)	`char password_type;`	password_type
-	`char password_dark;`	password_dark

mod [1]	**Datenstruktur kc_user_str**	**siehe** [2]
-	`char card_id[32];`[3] (nur auf BS2000-Systemen)	card_id
x(GPD)[4]	`char format_attr;` (nur auf BS2000-Systemen)	format_attr
x(GPD)[3]	`char format_name[7];` (nur auf BS2000-Systemen)	format_name
-	`char locale_lang_id[2];` (nur auf BS2000-Systemen)	locale_lang_id
-	`char locale_terr_id[2];` (nur auf BS2000-Systemen)	locale_terr_id
-	`char locale_ccsname[8];`	locale_ccsname
-	`char protect_pw_lth;`	protect_pw_lth
-	`char protect_pw_compl;`	protect_pw_compl
-	`char protect_pw_time[3];`	protect_pw_time
-	`char restart;`	restart
-	`char permit;`	permit
-	`char satsel;` (nur auf BS2000-Systemen)	satsel
-	`char user_type;`	user_type
-	`char lterm_curr[8];`	lterm_curr
-	`char connect_mode;`	connect_mode
-	`char in_service;`	in_service
-	`char number_tacs[10];`	number_tacs
-	`char cputime_sec[10];`	cputime_sec
-	`char seccounter[5];`	seccounter
-	`char deleted;`	deleted
x	`char protect_pw_time_left[3];`	protect_pw_time_left
-	`union kc_sign_date sign_time_date;`	sign_time_date
-	`char asyn_services[10];`	asyn_services
-	`char clients_signed[10];`	clients_signed
-	`char protect_pw_min_time[3];`	protect_pw_min_time
-	`char qlev[5];`	qlev

mod [1]	**Datenstruktur kc_user_str**	**siehe** [2]
-	`char out_queue[5];`	out_queue
x(GPD)	`char q_read_acl[8];`	q_read_acl
x(GPD)	`char q_write_acl[8];`	q_write_acl
-	`char q_mode;`	q_mode
-	`char certificate[10];` (nur auf BS2000-Systemen)	certificate
-	`char cert_auth[10];` (nur auf BS2000-Systemen)	cert_auth
x	`char pw_encrypted;`	pw_encrypted
x(GIR)	`char bcam_trace;`	bcam_trace
-	`char principal[100];` (nur auf BS2000-Systemen)	principal
-	`char node_last_excl_signon[4]`	node_last_excl_signon
-	`char exclusively_signed;`	exclusively_signed
-	`union kc_sign_date excl_sign_time_date;`	excl_sign_time_date
-	`char out_queue_ex[10];`	out_queue_ex
-	`char ptc;`	ptc
-	`char bound_ptc;`	bound_ptc
-	`char bound_service;`	bound_service
-	`char cputime_msec[10];`	cputime_msec
x(GPD)	`union kc_pw16 password16;`	password16
x(GPD)	`char protect_pw16_lth[2];`	protect_pw16_lth

1 Feldinhalt mit KC_MODIFY_OBJECT modifizierbar; siehe Abschnitt "obj_type=KC_USER"f

2 Die Bedeutung der Felder ist an der in dieser Spalte angegebenen Stelle beschrieben.

3 Ist standardmäßig mit Leerzeichen belegt

4 Beim Ändern des Startformats mit KC_MODIFY_OBJECT müssen Sie *format_name* **und** *format_attr* belegen.

Datenstruktur kc_user_fix_str	**siehe** [1]
`char us_name[8];`	us_name
`char card_position[3];` (nur auf BS2000-Systemen)	card_position
`char card_string_lth[3];` (nur auf BS2000-Systemen)	card_string_lth
`char card_string_type;` (nur auf BS2000-Systemen)	card_string_type
`union kc_string card_string;` (nur auf BS2000-Systemen)	card_string
`char card_id[32];` [2]	card_id
`char restart;`	restart
`char permit;`	permit
`char satsel;` (nur auf BS2000-Systemen)	satsel
`char user_type;`	user_type
`char qlev[5];`	qlev
`char certificate[10];` (nur auf BS2000-Systemen)	certificate
`char cert_auth[10];` (nur auf BS2000-Systemen)	cert_auth
`char principal[100];` (nur auf BS2000-Systemen)	principal

1 Die Bedeutung der Felder ist an der in dieser Spalte angegebenen Stelle beschrieben.

2 Ist standardmäßig mit Leerzeichen belegt

Datenstruktur kc_user_dyn1_str	**siehe** [1]
`char us_name[8];`	us_name
`char kset[8];`	kset
`char state;`	state
`char format_attr;` (nur auf BS2000-Systemen)	format_attr
`char format_name[7];` (nur auf BS2000-Systemen)	format_name
`char lterm_curr[8];`	lterm_curr
`char connect_mode;`	connect_mode
`char in_service;`	in_service
`char number_tacs[10];`	number_tacs
`char cputime_sec[10];`	cputime_sec
`char asyn_services[10];`	asyn_services
`char deleted;`	deleted
`char out_queue[10];`	out_queue
`char q_read_acl[8];`	q_read_acl
`char q_write_acl[8];`	q_write_acl
`char q_mode;`	q_mode
`char bcam_trace;`	bcam_trace
`char clients_signed[10];`	clients_signed
`union kc_sign_date sign_time_date`	sign_time_date
`char cputime_msec[10];`	cputime_msec

1 Die Bedeutung der Felder ist an der in dieser Spalte angegebenen Stelle beschrieben.

Datenstruktur kc_user_dyn2_str	**siehe** [1]
`char us_name[8];`	us_name
`union kc_pw password;`	password
`char password_type;`	password_type
`char password_dark;`	password_dark
`char locale_lang_id[2];` (nur auf BS2000-Systemen)	locale_lang_id
`char locale_terr_id[2];` (nur auf BS2000-Systemen)	locale_terr_id
`char locale_ccsname[8];` (nur auf BS2000-Systemen)	locale_ccsname
`char protect_pw_lth;`	protect_pw_lth
`char protect_pw_compl;`	protect_pw_compl
`char protect_pw_time[3];`	protect_pw_time
`char protect_pw_time_left[3];`	protect_pw_time_left
`char protect_pw_min_time[3];`	protect_pw_min_time
`char pw_encrypted;`	pw_encrypted
`char seccounter[5];`	seccounter
`char exclusively_signed;`	exclusively_signed
`union kc_sign_date excl_sign_time_date;`	excl_sign_time_date
`char node_last_excl_signon[4]`	node_last_excl_signon
`char ptc;`	ptc
`char bound_ptc;`	bound_ptc
`char bound_service;`	bound_service
`union kc_pw16 password16;`	password16
`char protect_pw16_lth[2]`	protect_pw16_lth

1 Die Bedeutung der Felder ist an der in dieser Spalte angegebenen Stelle beschrieben.

Die Felder der Datenstrukturen haben die folgende Bedeutung:

us_name

enthält den Namen der UTM-Benutzerkennung. Die Benutzerkennung gibt der Benutzer beim Anmelden bzw. ein UPIC-Client beim Aufbau einer Conversation mit der Anwendung an. *us_name* kann bis zu 8 Zeichen lang sein.

kset enthält den Namen des Keysets, das der Benutzerkennung zugeordnet ist. Das Keyset legt die Zugriffsrechte des Benutzers innerhalb der Anwendung fest. Der Benutzer kann einen Service nur dann aufrufen, wenn sowohl das Keyset der Benutzerkennung als auch das Keyset des LTERM-Partners (über den sich der Benutzer an der Anwendung anmeldet) einen Key- oder Zugangscode enthalten, der mit dem Lockcode bzw. der Access-List des angeforderten Services übereinstimmt.

Der Name eines Keysets kann bis zu 8 Zeichen lang sein.

Sie können in *kset* ein anderes Keyset definieren oder das aktuelle Keyset entfernen, indem Sie *kset* mit Leerzeichen füllen.

state gibt an, ob die Benutzerkennung zur Zeit zugelassen oder gesperrt ist.

'Y' Die Benutzerkennung ist zugelassen.

'N' Die Benutzerkennung ist zur Zeit gesperrt, es kann sich kein Benutzer oder Client mit dieser Benutzerkennung bei der Anwendung anmelden.

Die Benutzerkennung kann im laufenden Betrieb gesperrt bzw. wieder zugelassen werden. Ein Sperren wird beim nächsten Anmeldeversuch wirksam.

card_position
card_string_lth
card_string_type
card_string

Nur auf BS2000-Systemen: Diesen Feldern können Sie entnehmen, ob für den Zugang zur Anwendung über diese Benutzerkennung eine Magnetstreifenkarte nötig ist. Die Felder geben an, welches Teilfeld der Ausweisinformation auf der Magnetstreifenkarte geprüft wird und welche Informationen in diesem Teilfeld stehen müssen.

Die Angabe von *card_xx* schließt die Verwendung von *principal* aus.

card_position
gibt an, bei welchem Byte der Ausweisinformation die Prüfung beginnt; z.B. *card_position*='4' bedeutet, dass das 4. Byte der Ausweisinformation dem 1. Zeichen des zu prüfenden Teilfelds entspricht.

card_string_lth
gibt an, wie lang der zu prüfende Teil der Ausweisinformation ist. Die Länge wird in Byte angegeben.

card_string_type
gibt an, ob die zu prüfende Ausweisinformation als hexadezimale Zeichenfolge oder als Character-String zu interpretieren ist.

'X' Die Ausweisinformation ist eine hexadezimale Zeichenfolge.

'C' Die Ausweisinformation ist eine Zeichenfolge aus abdruckbaren, alphanumerischen Zeichen.

'N' Die Benutzerkennung wurde ohne Magnetstreifenkarte konfiguriert. In diesem Fall sind *card_string_lth* und *card_position* mit '0' belegt und in *card_string* werden Leerzeichen zurückgeliefert.

card_string
enthält die Zeichenfolge, die im zu prüfenden Teilbereich auf der Magnetstreifenkarte stehen muss, damit sich der Benutzer erfolgreich mit dieser Benutzerkennung bei der Anwendung anmelden kann.
UTM liefert die Zeichenfolge in einer Union des Typs *kc_string* zurück.

```
union kc_string
char x[200];
char c[100];
```

Ist die Ausweisinformation eine hexadezimale Zeichenfolge (*card_string_type* ='X'), wird jedes Halb-Byte als ein Zeichen dargestellt.
Ist *card_string_type* ='C', dann ist der Inhalt von *card_string* nach der in *card_string_lth* angegebenen Länge irrelevant.
Ist *card_string_type* ='X', dann ist der Inhalt von *card_string* nach der Länge 2 * *card_string_lth* irrelevant.

password

Dieser Parameter wird nicht mehr unterstützt.

password_type

gibt beim KC_GET_OBJECT-Aufruf an, ob für die Benutzerkennung ein Passwort generiert ist.

'Y' Für die Benutzerkennung ist ein Passwort generiert.

'N' Für die Benutzerkennung ist kein Passwort generiert.

Beim Ändern eines Passworts mit KC_MODIFY_OBJECT bzw. beim Neueintragen einer Benutzerkennung geben Sie in *password_type* an, wie das in *password* angegebene Passwort codiert ist.

'C' Das Passwort wird als Character-String angegeben.

'X' Das Passwort wird als hexadezimale Zeichenfolge angegeben.

Auf Unix-, Linux- und Windows-Systemen ist diese Angabe nur erlaubt, wenn das Passwort bereits verschlüsselt ist.

'N' Es wird kein Passwort angegeben.

password_dark

gibt an, ob das Passwort am Terminal dunkelgesteuert eingegeben werden muss:

'Y' UTM fordert den Benutzer beim Anmelden (KDCSIGN) in einem Zwischen-Dialog auf, das Passwort in ein dunkelgesteuertes Feld einzugeben.

'N' Der Benutzer muss das Passwort beim Anmelden (KDCSIGN) mit der Benutzerkennung an UTM übergeben. Das Passwort ist nicht dunkelgesteuert.

Nur für Unix- und Linux-Systeme:

Die Angabe in *password_dark* wird ignoriert. Das Passwort bekommt auf jeden Fall die Eigenschaft „dunkelgesteuert“. Ob das Passwort beim Anmelden über Dialog-Terminal-Prozesse in ein dunkelgesteuertes Feld eingegeben werden muss oder nicht, ist von der Generierung der Anwendung abhängig. Ist die Anwendung mit Formatierung generiert, muss das Passwort dunkelgesteuert eingegeben werden.

card_id

Nur auf BS2000-Systemen: Card-Identifier der Chipkarte.

Der Benutzer muss sich beim Anmelden mit Chipkarte identifizieren.

Beim Operationscode KC_GET_OBJECT werden Leerzeichen zurückgegeben, wenn der Benutzer ohne Chipkarte generiert ist.

format_attr
format_name

Nur auf BS2000-Systemen: Diese Felder beschreiben das Benutzer-spezifische Startformat. Dieses Startformat wird nach jedem erfolgreichen Anmelden automatisch am Terminal ausgegeben, wenn kein offener Vorgang für diese Benutzerkennung existiert. Befindet sich der Benutzer nach erfolgreicher Berechtigungsprüfung noch in einem Vorgang, so erscheint das Startformat nicht, stattdessen wird der letzte Dialog-Bildschirm ausgegeben (Vorgangswiederanlauf).

format_attr
enthält das Formatkennzeichen:

'A' (Formatattribut ATTR)
Das Startformat ist ein Format mit Benutzerattributen. Die Eigenschaften der Formatfelder können vom KDCS-Teilprogramm verändert werden. Der Formatname an der Programmschnittstelle KDCS ist + *formatname* .

'N' (Formatattribut NOATTR)

Das Startformat ist ein Format ohne Benutzerattribute. Weder Feld- noch Formateigenschaften können von KDCS-Teilprogrammen verändert werden. Der Formatname an der Programmschnittstelle KDCS ist * *formatname* .

'E' (Formatattribut EXTEND)
Das Startformat ist ein Format mit erweiterten Benutzerattributen. Es können sowohl die Eigenschaften der Formatfelder als auch globale Formateigenschaften von KDCS-Teilprogrammen verändert werden. Der Formatname an der Programmschnittstelle KDCS ist # *formatname* .

format_name
enthält den Namen des Startformats. Der Name kann bis zu 7 Zeichen lang sein und enthält nur alphanumerische Zeichen.

locale_lang_id
locale_terr_id
locale_ccsname

Nur auf BS2000-Systemen: Diese Felder enthalten die drei Komponenten des Locale, das der Benutzerkennung zugeordnet ist. Das Locale definiert die Sprachumgebung der Benutzer/Clients, die sich über Benutzerkennung anmelden (siehe auch openUTM-Handbuch „Anwendungen generieren“).

locale_lang_id
enthält das bis zu 2 Zeichen lange Sprachkennzeichen.

locale_terr_id
enthält ein bis zu 2 Zeichen langes Territorialkennzeichen.

locale_ccsname
(**c**oded **c**haracter **s**et **name**)
enthält den bis zu 8 Zeichen langen Namen eines erweiterten Zeichensatzes (CCS-Name; siehe auch Benutzerhandbuch zu XHCS).

protect_pw_lth

Dieser Parameter wird nicht mehr unterstützt.

protect_pw_compl

gibt an, welche Komplexitätsstufe das Passwort der Benutzerkennung haben muss.

'0' (NONE)
Es kann jede beliebige Zeichenfolge als Passwort angegeben werden.

'1' (MIN)
In dem Passwort dürfen maximal 2 aufeinanderfolgende Zeichen gleich sein.

'2' (MEDIUM)
In dem Passwort dürfen maximal 2 aufeinanderfolgende Zeichen gleich sein. Das Passwort muss mindestens einen Buchstaben und eine Ziffer enthalten.

'3' (MAX)
In dem Passwort dürfen maximal 2 aufeinanderfolgende Zeichen gleich sein. Das Passwort muss mindestens einen Buchstaben, eine Ziffer und ein Sonderzeichen enthalten. Sonderzeichen sind alle Zeichen, die von a-z, A-Z, 0-9 und Leerzeichen verschieden sind.

protect_pw_time

gibt an, wieviele Tage das Passwort maximal gültig ist (Gültigkeitsdauer).

Die Gültigkeit des Passworts läuft am Ende des letzten Tages der Gültigkeitsdauer ab. Wird z.B. eine Gültigkeit von einem Tag generiert, so läuft die Gültigkeit um 24:00 Uhr des folgenden Tages ab.

Kurz vor Ablauf der Gültigkeitsdauer fordert UTM den Benutzer mit der Meldung K121 auf, das Passwort zu ändern.

Ist die Gültigkeit abgelaufen, gilt Folgendes:
Ist das Grace-Signon generiert (*kc_signon_str.grace*='Y'), dann kann der Benutzer das Passwort bei der nächsten Anmeldung selbst ändern.

Ist das Grace-Signon nicht generiert, dann lehnt UTM eine Anmeldung mit der Meldung K120 ab. Der Administrator muss dann das Passwort ändern.

protect_pw_time='0' bedeutet, dass die Gültigkeitsdauer des Passworts nicht beschränkt ist.

restart gibt an, ob UTM für diese Benutzerkennung einen automatischen Vorgangswiederanlauf durchführt.

'Y' UTM führt für Benutzer, die sich unter der Benutzerkennung anmelden, einen automatischen Vorgangswiederanlauf durch.

UPIC-Clients, die unter der Benutzerkennung bei UTM angemeldet sind, können den Wiederanlauf eines offenen Vorgangs beim Aufbau einer neuen Verbindung initiieren, indem sie das Kommando KDCDISP absetzen.

'N' UTM führt für Benutzer, die sich unter der Benutzerkennung anmelden, keinen automatischen Vorgangswiederanlauf durch.

Ist die Anwendung mit SIGNON MULTI-SIGNON=YES generiert, dann können sich mehrere Benutzer/Clients gleichzeitig unter dieser Benutzerkennung anmelden. Dabei darf nur einer der Benutzer am Terminal angemeldet sein. Es können sich jedoch beliebig viele UPIC-Clients, TS-Anwendungen und OSI TP Partner gleichzeitig unter dieser Benutzerkennung anmelden.

permit gibt an, welche Berechtigung die Benutzerkennung innerhalb der lokalen Anwendung hat.

'A' (ADMIN)
Die Benutzerkennung hat Administrationsberechtigung, d.h. unter der Benutzerkennung dürfen alle Administrationsfunktionen in der lokalen Anwendung ausgeführt werden.

'N' (NONE)
Die Benutzerkennung hat keine Administrationsberechtigung.

Ist die lokale Anwendung eine UTM-Anwendung unter einem BS2000-System, dann dürfen unter dieser Benutzerkennung auch keine UTM-SAT-Administrationsfunktionen ausgeführt werden.

Nur auf BS2000-Systemen:

'B' (BOTH)
Unter der Benutzerkennung dürfen sowohl Administrations- als auch UTM-SAT-Administrationsfunktionen in der lokalen Anwendung ausgeführt werden.

'S' (SAT)
Die Benutzerkennung hat UTM-SAT-Administrationsberechtigung. Unter der Benutzerkennung dürfen Preselection-Funktionen ausgeführt werden, d.h. es kann die SAT-Protokollierung bestimmter Ereignisse ein- bzw. ausgeschaltet werden.

satsel

Nur auf BS2000-Systemen: gibt an, welche Ereignisse SAT für diese Benutzerkennung protokollieren soll (Benutzer-spezifische Einstellung). Voraussetzung für die Protokollierung ist, dass die SAT-Protokollierung für die Anwendung eingeschaltet ist (*kc_max_par_str.sat*='Y'). Zur SAT-Protokollierung siehe auch openUTM-Handbuch „Anwendungen generieren“ und openUTM-Handbuch „Einsatz von UTM-Anwendungen auf BS2000-Systemen“.

'B'

(BOTH)
Es werden erfolgreiche **und** nicht erfolgreiche Ereignisse protokolliert.

'S' (SUCCESS)
Es werden nur erfolgreiche Ereignisse protokolliert.

'F' (FAIL)
Es werden nur nicht erfolgreiche Ereignisse protokolliert.

'N' (NONE)
Es ist keine Benutzer-spezifische Art der SAT-Protokollierung definiert.

user_type

gibt bei Benutzerkennungen, die einem LTERM-Partner zugeordnet sind, den Typ des Client an, für den der LTERM-Partner erzeugt wurde.

'A' (APPLI)
Die Benutzerkennung ist dem LTERM-Partner einer TS-Anwendung vom Typ APPLI zugeordnet (PTERM mit PTYPE=APPLI).

'S' Die Benutzerkennung ist dem LTERM-Partner einer Socket-Anwendung zugeordnet (PTERM mit PTYPE= SOCKET).

'U' (UPIC)
Die Benutzerkennung ist dem LTERM-Partner eines UPIC-Clients zugeordnet (PTERM mit PTYPE=UPIC-R oder UPIC-L).

Bei allen anderen Benutzerkennungen wird in *user_type* ein Leerzeichen zurückgeliefert.

lterm_curr

Folgende Fälle sind zu unterscheiden:

Die Anwendung ist mit SIGNON MULTI-SIGNON=NO generiert
(d.h. Mehrfachanmeldungen nicht erlaubt):

lterm_curr enthält den LTERM- oder OSI-LPAP-Partner, über den ein Benutzer mit dieser Benutzerkennung angemeldet ist.

Ausnahme: *lterm_curr* enthält Leerzeichen, wenn die Anmeldung zum Starten eines Asynchron-Vorgangs über OSI TP erfolgte.

Die Anwendung ist mit SIGNON MULTI-SIGNON=YES generiert (Mehrfachanmeldungen erlaubt):

- Ist ein Benutzer mit der Benutzerkennung über ein Terminal mit der Anwendung verbunden, dann enthält *lterm_curr* den Namen des LTERM-Partners, der dem Terminal zugeordnet ist.
- Ist die Benutzerkennung mit *restart*='Y' generiert, dann enthält *lterm_curr* den Namen des LTERM- oder OSI-LPAP-Partners, über den ein Client mit dieser Benutzerkennung angemeldet ist.

  Ausnahmen: Die Anmeldung erfolgte über OSI TP und es wurde die Functional Unit „Commit“ ausgewählt oder die Anmeldung erfolgte über OSI TP zum Starten eines Asynchron-Vorgangs. Dann enthält *lterm_curr* Leerzeichen.

In allen anderen Fällen enthält *lterm_curr* Leerzeichen.

connect_mode

gibt an, ob z.Zt. ein Benutzer oder Client mit dieser Benutzerkennung über den LTERM- bzw. OSI-LPAP-Partner in *lterm_curr* angemeldet ist ('Y') oder nicht ('N').

in_service

gibt an, ob über den LTERM- bzw. OSI-LPAP-Partner in *lterm_curr* unter dieser Benutzerkennung zur Zeit ein Vorgang bearbeitet wird.

'Y' Es ist ein Vorgang offen, der mindestens einen Sicherungspunkt erreicht hat.

'N' Es ist zur Zeit kein Vorgang offen, der mindestens einen Sicherungspunkt erreicht hat.

number_tacs

enthält die Anzahl der unter dieser Benutzerkennung ausgeführten Teilprogrammläufe. In UTM-S-Anwendungen wird der Wert von *number_tacs* bei jeder Neugenerierung mit KDCDEF oder Änderungsgenerierung mit KDCDEF/KDCUPD auf 0 gesetzt. Bei UTM-F-Anwendungen wird *number_tacs* bei jedem Anwendungsstart auf 0 gesetzt.

cputime_sec

enthält die Anzahl CPU-Sekunden, die für die Bearbeitung der Aufträge für diese Benutzerkennung seit dem letzten Verbindungsaufbau verbraucht wurden. Der in *cputime_sec* zurückgelieferte Wert enthält jedoch nicht die für Datenbankaufrufe verbrauchte CPU-Zeit.

seccounter

enthält die Anzahl der Sicherheitsverletzungen für diese Benutzerkennung (z.B. falsches Passwort eingegeben, nicht erlaubte Transaktionscodes aufgerufen) seit dem Start der Anwendung.

deleted

gibt an, ob die Benutzerkennung aus der Konfiguration gelöscht wurde oder nicht.

'Y' Die Benutzerkennung wurde verzögert gelöscht (KC_DELAY). Der Name ist aber gesperrt, d.h. es kann keine neue Benutzerkennung mit diesem Namen erzeugt werden.

'N' Die Benutzerkennung wurde nicht gelöscht.

protect_pw_time_left

Beim Opcode KC_GET_OBJECT:
gibt die Zeit in Tagen an, wie lange das Passwort noch gültig ist.

Darüber hinaus sind folgende Werte möglich:

' ' (Leerzeichen) Für die Benutzerkennung wurde kein Passwort generiert oder das Passwort wurde gelöscht.

'000' Das Passwort läuft noch am aktuellen Tag ab.

'-1' Der Benutzerkennung wurde ein Passwort mit unbeschränkter Gültigkeitsdauer zugeordnet (*protect_pw_time* ='0').

'-2' Die Gültigkeitsdauer des Passworts ist bereits abgelaufen.

Beim Opcode KC_MODIFY_OBJECT:
Nur relevant in Anwendungen, die mit SIGNON GRACE=YES generiert sind, und für Benutzerkennungen, für die eine begrenzte Gültigkeitsdauer des Passworts generiert ist.
In *protect_pw_time_left* geben Sie an, ob für das neue Passwort die generierte Gültigkeitsdauer wirksam sein soll. Eine Angabe in diesem Feld wird nur berücksichtigt zusammen mit einer Angabe in *password* und *password_type*.
Wenn Sie *protect_pw_time*='-1' (rechts- oder linksbündig) angeben, ist die generierte Gültigkeitsdauer (vom Zeitpunkt der Änderung an) für das neue Passwort wirksam.Wenn Sie nichts angeben, ist das neue Passwort wegen Ablauf der Gültigkeitsdauer sofort ungültig. Der Benutzer muss beim nächsten Anmelden sein Passwort ändern.
Ein anderer Wert als '-1' wird zurückgewiesen.

sign_time_date

gibt an, wann sich zuletzt ein Benutzer oder Client mit dieser Benutzerkennung bei UTM angemeldet hat.

UTM liefert Datum und Uhrzeit der letzten Anmeldung im Feld *cstring* einer Union des Typs *kc_sign _date* zurück.

union kc_sign_date

```
char cstring[14];
struct cstr_str cstring_struct;
```

wobei

struct cstr_str

```
char year[4];
char month [2];
char day[2];
char hour[2];
char minute[2];
char sec[2];
```

Die Ausgabe erfolgt in der Form 'YYYYMMDDhhmmss'. Dabei ist *YYYY* das Jahr, *MM* der Monat, *DD* der Tag, *hh* die Stunde, *mm* die Minute und *ss* die Sekunde.

Wenn sich mit der Benutzerkennung bisher noch kein Benutzer oder Client bei der Anwendung angemeldet hat, liefert UTM '00000000000000' zurück.

asyn_services

enthält die Anzahl der zur Zeit für diese Benutzerkennung laufenden Asynchron-Aufträge.

clients_signed

enthält die Anzahl der Kommunikationspartner, die aktuell unter dieser Benutzerkennung an die Anwendung angemeldet sind.

Auch in Anwendungen, die mit SIGNON MULTI-SIGNON=NO generiert wurden, kann der Wert kurzfristig größer als 1 sein, wenn sich aktuell ein OSI TP-Kommunikationspartner unter dieser Benutzerkennung zum Erzeugen eines Asynchron-Auftrags angemeldet hat.

protect_pw_min_time

gibt die minimale Gültigkeitsdauer des Passworts in Tagen an.
Nach der Änderung des Passworts darf der Benutzer das Passwort frühestens nach Ablauf der minimalen Gültigkeitsdauer erneut ändern.

Nach einer Passwortänderung durch den Administrator bzw. nach einer Neugenerierung kann der Benutzer das Passwort immer ändern, unabhängig davon, ob die minimale Gültigkeitsdauer abgelaufen ist oder nicht.

qlev

(**Q**ueue **Lev**el) zeigt an, wieviele Nachrichten in der Queue des Benutzers maximal zwischengespeichert werden können. Wird der Schwellwert überschritten, dann hängt das weitere Verhalten von UTM vom Wert im Feld *q_mode* ab.

UTM berücksichtigt die für die Queue erzeugten Nachrichten erst am Ende der Transaktion. Daher kann die in *qlev* festgelegte Anzahl von Nachrichten für eine Message Queue überschritten werden, wenn in einer Transaktion mehrere Nachrichten für dieselbe Queue erzeugt wurden.

Bei *qlev*=0 können keine Nachrichten in der Queue gespeichert werden, bei *qlev*=32767 ist die Queue-Länge nicht begrenzt.

out_queue

zeigt die Anzahl der Nachrichten in der Nachrichten-Queue des Benutzers an.

Detaillierte Information dazu entnehmen Sie dem openUTM-Handbuch „Anwendungen generieren“.

Ist die Anzahl der Nachrichten größer als 99999, wird die Zahl nur unvollständig dargestellt. Deshalb sollte das Feld *out_queue_ex* bzw. das Feld *out_queue* aus der Datenstruktur *kc_user_dyn1* verwendet werden, weil hier auch größere Zahlen vollständig abgebildet werden.

q_read_acl

zeigt die Rechte an (Name eines Keysets), die ein fremder Benutzer benötigt, um Nachrichten aus der User-Queue lesen und löschen zu können.

Ein fremder Benutzer kann nur dann lesend auf diese Queue zugreifen, wenn das Keyset seiner Benutzerkennung und das Keyset des LTERM-Partners, über den der Benutzer angemeldet ist, jeweils mindestens einen Keycode enthalten, der auch in dem angezeigten Keyset enthalten ist.

Enthält *q_read_acl* keinen Wert, dann können alle Benutzer Nachrichten aus dieser Queue lesen und löschen.

q_write_acl

zeigt die Rechte an (Name eines Keysets), die ein fremder Benutzer benötigt, um Nachrichten in die User-Queue zu schreiben.

Ein fremder Benutzer kann nur dann schreibend auf diese Queue zugreifen, wenn das Keyset seiner Benutzerkennung und das Keyset des LTERM-Partners, über den der Benutzer angemeldet ist, jeweils mindestens einen Keycode enthalten, der auch in dem angezeigten Keyset enthalten ist.

Enthält *q_write_acl* keinen Wert, dann können alle Benutzer Nachrichten in diese Queue schreiben.

q_mode

(**Q**ueue **Mode**) zeigt an, wie sich UTM verhält, wenn die maximale Anzahl der noch nicht ausgeführten Aufträge in der Queue des Benutzers erreicht ist (siehe *qlev*). Mögliche Werte sind:

'S' UTM lehnt weitere Nachrichten ab.

'W' UTM nimmt weitere Nachrichten auf. Beim Schreiben einer neuen Nachricht in die Queue wird dann aber die älteste in der Queue stehende Nachricht gelöscht.

certificate

Nur auf BS2000-Systemen: Dieser Parameter wird nicht mehr unterstützt.

cert_auth

Nur auf BS2000-Systemen: Dieser Parameter wird nicht mehr unterstützt

pw_encrypted

Das Feld *pw_encrypted* ist nur bei KC_MODIFY_OBJECT relevant. Bei der Informationsabfrage mit KC_GET_OBJECT ist *pw_encrypted* immer mit Leerzeichen belegt.

Beim Ändern des Passworts geben Sie im Feld *pw_encrypted* an, ob das in *password16* angegebene Passwort schon verschlüsselt ist.

'N' Das Passwort ist unverschlüsselt (Standard).

'Y' /'A' Das Passwort ist schon verschlüsselt. Dies kann z.B. dann der Fall sein, wenn das verschlüsselte Passwort aus der Meldung K159 in einer Standby-Anwendung stammt.

bcam_trace

gibt an, ob der BCAM-Trace für diesen USER explizit eingeschaltet ist.

'Y' Der BCAM-Trace ist für diesen USER explizit eingeschaltet.

'N' Der BCAM-Trace ist für diesen USER nicht explizit eingeschaltet.

Die Auswertung des Feldes mit KC_GET_OBJECT ist nur dann sinnvoll, wenn der BCAM-Trace für einzelne USER explizit eingeschaltet ist. Ist der BCAM-Trace allgemein eingeschaltet (siehe *kc_diag_and_account_par_str*), wird hier für diesen USER *bcam_trace*='N' zurückgeliefert.

Beim Aufruf von KC_MODIFY_OBJECT kann der BCAM-Trace explizit ein- oder ausgeschaltet werden. Der BCAM-Trace kann nur dann für einen einzelnen USER eingeschaltet werden,

- wenn er für alle USER ausgeschaltet ist (siehe *kc_diag_and_account_par_str*) oder

- wenn er bisher nur für einzelne USER eingeschaltet ist.

principal

Nur auf BS2000-Systemen: Die Authentisierung des Benutzers erfolgt über Kerberos. Die Authentisierung über Kerberos ist nur möglich, wenn die Anmeldung des Benutzers direkt (nicht über OMNIS) von einem Kerberos-fähigen Terminal aus erfolgt.

Die Angabe von *principal* schließt die Verwendung von *card_xx* und *password* aus.

Bei der Abfrage mit KC_GET_OBJECT wird hier der Principal angezeigt, falls der Benutzer mit Kerberos-Authentifizierung generiert ist.

Beim Aufruf von KC_CREATE_OBJECT geben Sie hier eine alphanumerische Zeichenfolge in folgender Form an:

`windowsaccount@NT-DNS-REALM-NAME'`

windowsaccount

Domänen-Kennung des Benutzers

NT-DNS-REALM-NAME

DNS-Name der Active-Directory-Domäne. Dieser Name ist ein fester Wert für jede Active-Directory-Domäne, der schon beim Einrichten des Kerberos-Schlüssels vergeben wurde.

node_last_excl_signon

Dieses Feld ist nur für UTM-Cluster-Anwendungen relevant.
Nummer (Index) der Knoten-Anwendung, bei der zuletzt ein Benutzer/Client mit dieser Benutzerkennung exklusiv angemeldet war.

exclusively_signed

Dieses Feld ist nur für UTM-Cluster-Anwendungen relevant.
exclusively_signed gibt an, ob ein Benutzer/Client aktuell mit dieser Benutzerkennung exklusiv angemeldet ist.

'Y' Der Benutzer/Client ist zur Zeit exklusiv angemeldet.

'N' Es ist kein Benutzer/Client mit der Benutzerkennung exklusiv angemeldet.

excl_sign_time_date

Dieses Feld ist nur für UTM-Cluster-Anwendungen relevant.
Datum und Uhrzeit der letzten exklusiven Anmeldung dieses Benutzers.

UTM liefert Datum und Uhrzeit der letzten Anmeldung in einer Union des Typs *kc_sign_date* zurück.

union kc_sign_date

```
char cstring[14];
struct cstr_str cstring_struct;
```

wobei

struct cstr_str
`char year[4];`
`char month [2];`
`char day[2];`
`char hour[2];`
`char minute[2];`
`char sec[2];`

Die Ausgabe erfolgt in der Form 'YYYYMMDDhhmmss'. Dabei ist *YYYY* das Jahr, *MM* der Monat, *DD* der Tag, *hh* die Stunde, *mm* die Minute und *ss* die Sekunde. Wenn sich mit der Benutzerkennung bisher noch kein Benutzer oder Client bei der Anwendung exklusiv angemeldet hat, liefert openUTM '00000000000000' zurück.

out_queue_ex

siehe out_queue.

ptc Der Benutzer hat einen offenen Vorgang mit einer Transaktion im Zustand PTC.

bound_ptc

Der Benutzer hat einen knotengebundenen Vorgang mit einer Transaktion im Zustand PTC (nur relevant für UTM-Cluster-Anwendungen).

bound_service

Der Benutzer hatte beim letzten Abmelden einen knotengebundenen Vorgang (nur relevant für UTM-Cluster-Anwendungen).

cputime_msec

enthält die Anzahl CPU-Millisekunden, die für die Bearbeitung der Aufträge für diese Benutzerkennung seit dem letzten Verbindungsaufbau verbraucht wurden. Der in *cputime_msec* zurückgelieferte Wert enthält jedoch nicht die für Datenbankaufrufe verbrauchte CPU-Zeit.

password16

ist bei der Informationsabfrage mit KC_GET_OBJECT immer mit Leerzeichen belegt, auch wenn für die Benutzerkennung ein Passwort definiert ist.

Das Feld *password16* ist nur relevant bei KC_MODIFY_OBJECT und KC_CREA-TE_OBJECT. Dann können Sie in *password16* das neue Passwort der Benutzerkennung an UTM übergeben (siehe Abschnitt

"obj_type = KC_USER" im Kapitel "KC_CREATE_OBJECT - Objekte in die Konfiguration eintragen" und Abschnitt "obj_type = KC_USER" im Kapitel "KC_MODIFY_OBJECT - Objekteigenschaften und Anwendungsparameter ändern").

Auf BS2000-Systemen schließt die Angabe von *password16* die Verwendung von *principal* aus.

protect_pw16_lth

gibt an, wieviele Zeichen ein Passwort der Benutzerkennung mindestens haben muss, damit es von UTM akzeptiert wird (minimale Länge des Passworts). Der Administrator kann das Passwort eines Benutzers nur löschen, wenn in *protect_pw16_lth* '00' zurückgeliefert wird.

11.3.2 Datenstrukturen zur Beschreibung der Anwendungsparameter

Im Folgenden werden alle Datenstrukturen beschrieben, die für die Übergabe der Anwendungsparameter zur Verfügung stehen. Für jeden einzelnen Parametertyp steht in der Include-Datei *kcadminc.h* eine eigene Datenstruktur zur Verfügung. Der Name der jeweiligen Datenstruktur setzt sich zusammen aus dem Namen des Parametertyps und dem Suffix „_str“. Die Beschreibung erfolgt in alphabetisch aufsteigender Reihenfolge der Datenstrukturnamen.

11.3.2.1 kc_cluster_curr_par_str - Statistikwerte einer UTM-Cluster-Anwendung

Für den Objekttyp KC_CLUSTER_CURR_PAR ist die Datenstruktur *kc_cluster_curr_par_str* definiert. In *kc_cluster_curr_par_str* liefert UTM bei KC_GET_OBJECT die Informationen zur Belegung des Cluster Pagepools zurück.

Mit KC_MODIFY_OBJECT können Zähler auf 0 zurückgesetzt werden.

mod[1]	**Datenstruktur kc_cluster_curr_par_str**
x(GID)	`char max_cpgpool_size[3];`
-	`char curr_cpgpool_size[3];`
x(GID)	`char avg_cpgpool_size[3];`
-	`char node_reserved_cpgpool_pages[10];`

1 Feldinhalt mit KC_MODIFY_OBJECT modifizierbar; siehe Abschnitt "obj_type = KC_CLUSTER_CURR_PAR"

Die Felder der Datenstruktur haben die folgende Bedeutung:

max_cpgpool_size

gibt die maximale Belegung des Cluster Pagepools in % an.

Der Wert gilt über den Lauf der gesamten UTM-Cluster-Anwendung hinaus. Er wird bei Vergrößerung des Cluster Pagepools und beim Erzeugen der UTM-Cluster-Dateien per KDCDEF zurückgesetzt.

KC_MODIFY_OBJECT:
Setzt den Wert auf 0 zurück. Damit wird implizit auch der Wert von *avg_cpgpool_size* auf 0 zurückgesetzt.

curr_cpgpool_size

gibt die aktuelle Belegung des Cluster Pagepools in % an.

avg_cpgpool_size

gibt die durchschnittliche Belegung des Cluster Pagepools in % an.

Der Wert gilt über den Lauf der gesamten UTM-Cluster-Anwendung hinaus. Er wird bei Vergrößerung des Cluster Pagepools und beim Erzeugen der UTM-Cluster-Dateien per KDCDEF zurückgesetzt.

KC_MODIFY_OBJECT:
Setzt den Wert auf 0 zurück. Damit wird implizit auch der Wert von *max_cpgpool_size* auf 0 zurückgesetzt.

node_reserved_cpgpool_pages

gibt die Anzahl reservierter Seiten des aktuellen lokalen Knotens an.

11.3.2.2 kc_cluster_par_str - Globale Eigenschaften einer UTM-Cluster-Anwendung

Für den Parametertyp KC_CLUSTER_PAR ist die Datenstruktur *kc_cluster_par_str* definiert. In *kc_cluster_par_str* liefert UTM bei KC_GET_OBJECT die aktuellen Einstellungen für die Eigenschaften einer UTM-Cluster-Anwendung und aktuelle Daten (z.B. Generierungs-, Startzeitpunkt, Anzahl der aktiven und der generierten Knoten-Anwendungen) zurück.Mit KC_MODIFY_OBJECT können Sie Folgendes anpassen:

- Parameter, die die Prüfung der Verfügbarkeit der einzelnen Knoten-Anwendungen regeln
- die Parameter, die den Zugriff der Knoten-Anwendungen auf die Cluster-Konfigurationsdatei und das Cluster-Administrations-Journal regeln

mod [1]	Datenstruktur kc_cluster_par_str
-	`struct kc_cluster_filebase cluster_filebase;`
-	`struct kc_admi_date_time_model gen_time;`
-	`char os_type[24];`
-	`char bit_mode[8];`
-	`char bcamappl[8];`
-	`char port_nbr[8];`
x(GID)	`char check_alive_timer_sec[8];`
x(GID)	`char communication_retry[8];`
x(GID)	`char communication_reply_timer_sec[8];`
x(GID)	`char restart_timer_sec[8]:`
x(GID)	`char file_lock_timer_sec[8];`
x(GID)	`char file_lock_retry[8];`
-	`char max_nbr_nodes[4];`
-	`char curr_nbr_nodes[4];`
-	`char nbr_active_nodes[4];`
-	`char emergency_cmd [200];`
-	`char failure_cmd [200];`
-	`struct kc_admi_date_time_model last_kdcdef_time;`
-	`struct kc_admi_date_time_model cluster_start_time;`
-	`char abort_bound_service;`
x(GID)	`char deadlock_prevention;`
-	`char listener_id[5];` (nur auf Unix-, Linux- und Windows-Systemen)
-	`char cpgpool[10];`
-	`char cpgpool_warnlevel[2];`
-	`char cpgpool_fs[2];`

1 Feldinhalt mit KC_MODIFY_OBJECT modifizierbar; siehe Abschnitt "obj_type=KC_CLUSTER_PAR"

Die Felder der Datenstruktur *kc_cluster_par_str* entsprechen den Konfigurationsinformationen der KDCDEF-Steueranweisung CLUSTER, siehe openUTM-Handbuch „Anwendungen generieren“.

Die Felder der Datenstruktur haben die folgende Bedeutung:

cluster_filebase

Namens-Präfix bzw. Dateiverzeichnis (Basisname) der Cluster-Konfigurationsdatei und anderer Verwaltungsdateien der UTM-Cluster-Anwendung, wie z.B. des Cluster-Administrations-Journals. Der Name wird in dem Element *cluster_filebase* vom Typ *kc_cluster_filebase* übergeben:

struct kc_cluster_filebase
`char length[2];` `char fb_name[54];`

fb_name enthält den Basisnamen, *length* die Länge des Basisnamens.

gen_time

Zeitpunkt zu dem die Cluster-Konfigurationsdatei erzeugt wurde (Zeitpunkt der Generierung). Datum und Uhrzeit werden in dem Element *gen_time* vom Typ *kc_admi_date_time_model* zurückgeliefert.

struct kc_admi_date_time_model
`struct kc_admi_date_model admi_date;` `struct kc_admi_time_model admi_time`

wobei

struct kc_admi_date_model
`char admi_day [2];`
`char admi_month [2];`
`char admi_year_4 [4];`
`char admi_julian_day [3];`
`char admi_daylight_saving_time`

und

struct kc_admi_time_model
`char admi_hours [2];`
`char admi_minutes [2];`
`char admi_seconds [2]`

os_type

Systemplattform des Rechners, z.B. 'Solaris Sparc'.

bit_mode

Modus, in dem das Betriebssystem abläuft. Zurückgeliefert wird:

'32 Bit' für den 32-Bit-Modus.

'64 Bit' für den 64-Bit-Modus.

bcamappl

Name des Transportsystemendpunkts (BCAMAPPL-Name), der für die Cluster-interne Kommunikation verwendet wird.

port_nbr

Nummer des Listener-Ports, der für die Cluster-interne Kommunikation verwendet wird.

check_alive_timer_sec

In einer UTM-Cluster-Anwendung wird jede Knoten-Anwendung durch eine andere Knoten-Anwendungen überwacht (Ringüberwachung), d.h. jede Knoten-Anwendung überprüft die Verfügbarkeit

einer anderen Knoten-Anwendung und wird selbst von einer Knoten-Anwendung überwacht. Dazu schickt die überwachende Knoten-Anwendung in bestimmten Zeitintervallen (*check_alive_timer_sec*) Nachrichten an die zu überwachende Knoten-Anwendung. Ist diese verfügbar, quittiert diese die Nachricht.
check_alive_timer_sec gibt den Zeitabstand in Sekunden an, in dem Überwachungs-Nachrichten an die zu überwachende Knoten-Anwendung geschickt werden. Dieser Timer wird außerdem zum periodischen Zugriff auf die Cluster-Konfigurationsdatei und das Cluster-Administrations-Journal verwendet.

KC_MODIFY_OBJECT:
Sie können das Überwachungsintervall ändern.

Minimalwert:'30'
Maximalwert:'3600'

communication_retry

gibt an, wie oft eine Knoten-Anwendung erneut versucht, eine Überwachungs-Nachricht zu senden, wenn die zu überwachende Knoten-Anwendung nicht innerhalb der in *communication_reply_timer_sec* festgesetzten Zeit antwortet. Antwortet die zu überwachende Knoten-Anwendung auf keinen der Wiederholungsversuche in der festgesetzen Zeit, wird ihr Ausfall angenommen und die in *failure_cmd* definierte Kommandofolge ausgeführt (z.B. ein Neustart).

KC_MODIFY_OBJECT:
Sie können den Wert von *communication_retry* ändern.

Minimalwert:'0'
Maximalwert:'10'

communication_reply_timer_sec

Zeit in Sekunden, die eine Knoten-Anwendung nach dem Senden einer Überwachungs-Nachricht maximal auf Antwort wartet.
Trifft in dieser Zeit keine Antwort ein, wird der Ausfall (abnormales Anwendungsende) der zu überwachenden Knoten-Anwendung angenommen und die in *failure_cmd* definierte Kommandofolge ausgeführt (z.B. ein Neustart).

Ist für *communication_retry* ein Wert größer Null gesetzt, wird erst dann von einem Ausfall der anderen Knoten-Anwendung ausgegangen, wenn eine Antwort auf die Überwachungs-Nachricht auch nach dem letzten Wiederholversuch ausbleibt.

KC_MODIFY_OBJECT:
Sie können die Einstellung für *communication_reply_timer_sec* ändern.

Minimalwert: '1'
Maximalwert: '60'

restart_timer_sec

Zeit in Sekunden, die eine Knoten-Anwendung nach einem Ausfall (abnormales Programmende) maximal für einen Warmstart benötigt.

Die überwachende Knoten-Anwendung wartet nach dem Aufruf der in *failure_cmd* festgesetzten Kommandofolge für die hier angegebene Zeit, bis sie wieder eine Überwachungs-Nachricht an diese Knoten-Anwendung sendet. Erhält die überwachende Knoten-Anwendung auf diese Nachricht keine

Antwort, wird angenommen, dass die ausgefallene Knoten-Anwendung aufgrund eines permanenten Problems nicht mehr gestartet werden kann. Es wird die in *emergency_cmd* festgesetzte Kommanofolge für die ausgefallene Knoten-Anwendung aufgerufen.

KC_MODIFY_OBJECT:
Sie können den Wert von *restart_timer_sec* ändern.

Minimalwert: '0', d.h. keine Zeitüberwachung des Neustarts
Maximalwert: '3600'

file_lock_timer_sec

Zeit in Sekunden, die eine Knoten-Anwendung maximal auf die Zuteilung einer Sperre für den Zugriff auf die Cluster-Konfigurationsdatei oder das Cluster-Administrations-Journal wartet.
file_lock_retry gibt an, wie oft eine Knoten-Anwendung die Anforderung einer Sperre für die Cluster-Konfigurationsdatei oder das Cluster-Administrations-Journal wiederholt, wenn die Sperre nicht in der in *file_lock_timer_sec* vorgegebenen Zeit zugeteilt werden konnte.

KC_MODIFY_OBJECT:
Setzt einen neuen Wert für file_lock_timer_sec.

Minimalwert:'10'
Maximalwert:'60'

file_lock_retry

gibt an, wie oft eine Knoten-Anwendung die Anforderung einer Sperre für die Cluster-Konfigurationsdatei oder das Cluster-Administrations-Journal wiederholt, wenn die Sperre nicht in der in *file_lock_timer_sec* vorgegebenen Zeit zugeteilt werden konnte.

KC_MODIFY_OBJECT:
Sie können den Wert von *file_lock_retry*. ändern:

Minimalwert:'1'
Maximalwert:'10'

max_nbr_nodes

Maximal mögliche Anzahl von Knoten-Anwendungen, die in einer UTM-Cluster-Anwendung generiert werden können.

Im XCS-Verbund von BS2000-Systemen können von den 32 generierbaren Knoten-Anwendungen immer nur maximal 16 Knoten-Anwendungen gleichzeitig laufen.

curr_nbr_nodes

Anzahl der für diese UTM-Cluster-Anwendung tatsächlich generierten Knoten-Anwendungen (entspricht der Anzahl der CLUSTER-NODE-Anweisungen bei der KDCDEF-Generierung der UTM-Cluster-Anwendung).

nbr_active_nodes

Anzahl der zur Zeit in der UTM-Cluster-Anwendung aktiven (gestarteten) Knoten-Anwendungen.

emergency_cmd

Enthält ein auszuführendes Kommando und dessen Aufrufparameter.
Dieses Kommando wird von UTM aufgerufen, wenn eine ausgefallene Knoten-Anwendung nicht neu

gestartet werden kann und für *restart_timer_sec* ein Wert größer Null eingestellt ist. D.h. die in *failure_cmd* angegebenen Aktionen haben nicht dazu geführt, dass die ausgefallene Knoten-Anwendung wieder (rechtzeitig) gestartet wurde.

failure_cmd

Enthält ein auszuführendes Kommando und dessen Aufrufparameter. Dieses Kommando wird von UTM aufgerufen, wenn eine Knoten-Anwendung abnormal beendet oder der Ausfall einer Knoten-Anwendung erkannt worden ist. Mit dem Kommando in *failure_cmd* kann z.B. ein Neustart der ausgefallenen Knoten-Anwendung veranlasst werden oder eine E-Mail an den Systemverwalter geschickt werden.

last_kdcdef_time

Zeitpunkt der letzten Generierung einer KDCFILE, mit der bereits ein Start zumindest einer Knoten-Anwendung erfolgt ist.

Datum und Uhrzeit werden in dem Element *last_kdcdef_time* vom Typ *kc_admi_date_time_model* zurückgeliefert (siehe *gen_time*).

cluster_start_time

Zeitpunkt, zu dem die erste Knoten-Anwendung der UTM-Cluster-Anwendung gestartet wurde.

Datum und Uhrzeit des Starts werden in dem Element *cluster_start_time* vom Typ *kc_admi_date_time_model* zurückgeliefert (siehe *gen_time*).

abort_bound_service

'N' Gibt es beim Anmelden für einen Benutzer einen offenen Vorgang, der an eine andere Knoten-Anwendung gebunden ist, dann ist ein Anmelden nur an der Knoten-Anwendung möglich, an die der offene Vorgang gebunden ist. Die Anmeldung an jeder anderen Knoten-Anwendung wird abgelehnt.

'Y' Meldet sich ein Benutzer an eine Knoten-Anwendung an und gibt es für den Benutzer einen offenen Vorgang, der an eine andere Knoten-Anwendung gebunden ist, welche beendet wurde, dann kann sich der Benutzer anmelden, falls keine Transaktion des offenen Vorgangs im Zustand PTC ist. Ein Vorgangswiederanlauf findet dabei nicht statt.

Der offene Vorgang wird beim nächsten Start der Knoten-Anwendung, an die er gebunden ist, abnormal beendet.

deadlock_prevention

gibt an, ob UTM für die Datenbereiche GSSB, TLS und ULS zusätzliche Prüfungen zur Deadlock-Vermeidung durchführt oder nicht.

'N' UTM führt für die Datenbereiche GSSB, TLS und ULS keine zusätzlichen Prüfungen zur Deadlock-Vermeidung durch. Kommt es zu einem Deadlock auf diesen Datenbereichen, dann wird dieser über einen Timeout aufgelöst.

'Y' UTM führt für die Datenbereiche GSSB, TLS und ULS zusätzliche Prüfungen zur Deadlock-Vermeidung durch.

Es wird empfohlen, diesen Parameter im Produktivbetrieb nur dann auf 'Y' zu setzen, wenn es häufig zu Timeouts beim Zugriff auf diese Datenbereiche kommt.

listener_id (nur auf Unix-, Linux- und Windows-Systemen)

Dieser Parameter dient dazu, einen Netzprozess für die Cluster-interne Kommunikation auszuwählen.

cpgpool

Größe des Cluster Pagepools in 4K-Seiten.

cpgpool_warnlevel

Prozentwert, der angibt, bei welcher Belegung des Cluster Pagepools eine Warnung (Meldung K041) ausgegeben wird.

cpgpool_fs

Anzahl der Dateien, auf die die Anwenderdaten im Cluster Pagepool aufgeteilt sind.

11.3.2.3 kc_curr_par_str - Aktuelle Werte der Anwendungsparameter

Für den Parametertyp KC_CURR_PAR ist die Datenstruktur *kc_curr_par_str* definiert.
In *kc_curr_par_str* liefert UTM bei KC_GET_OBJECT die aktuell eingestellten Parameterwerte, Daten zum Anwendungslauf und Statistikinformationen zur Auslastung der Anwendung zurück (siehe auch KDCINF im Abschnitt „type=STATISTICS" (Ausgabe von KDCINF (Beispiele))).

Mit KC_MODIFY_OBJECT können Sie einige Zählerstände, die UTM zur Erstellung von Statistikinformationen verwendet, bei Bedarf auf 0 zurücksetzen (siehe auch *max_statistics_msg* im Abschnitt "kc_max_par_str - Maximalwerte der Anwendung (MAX-Parameter)").

Bei MAX STATISTICS-MSG=NONE werden die Zählerstände in einer UTM-S-Anwendung nur beim ersten Start der Anwendung und in UTM-F-Anwendungen bei jedem Anwendungsstart zurückgesetzt.

Bei MAX STATISTICS-MSG=FULL-HOUR werden die Zählerstände zu jeder vollen Stunde zurückgesetzt. Deshalb können in dem ersten Intervall nach einer vollen Stunde zu kleine Werte angezeigt werden.

mod [1]	Datenstruktur kc_curr_par_str
-	char appliname[8];
-	char utm_version[8];
-	char applimode;
-	char start_date_year[4];
-	char start_date_month[2];
-	char start_date_day[2];
-	char start_time_hour[2];
-	char start_time_min[2];
-	char start_time_sec[2];
-	char curr_date_year[4];
-	char curr_date_month[2];
-	char curr_date_day[2];
-	char curr_time_hour[2];
-	char curr_time_min[2];
-	char curr_time_sec[2];
x(GIR)	char term_input_msgs[10];
x(GIR)	char term_output_msgs[10];
-	char curr_max_asyntasks[3];
-	char curr_max_tasks_in_pgwt[3];
-	char curr_tasks[3];
-	char curr_asyntasks[3];
-	char curr_tasks_in_pgwt[3];
-	char tasks_waiting_in_pgwt[3];
-	char connected_users[10];
-	char open_dial_services[10];
-	char open_asyn_services[10];

mod [1]	Datenstruktur kc_curr_par_str
-	char dial_ta_per_100sec[10];
-	char asyn_ta_per_100sec[10];
-	char dial_step_per_100sec[10];
x(GIR)	char max_dial_ta_per_100sec[10];
x(GIR)	char max_asyn_ta_per_100sec[10];
x(GIR)	char max_dial_step_per_100sec[10];
x(GIR)	char max_pool_size[3];
-	char curr_pool_size[3];
x(GIR)	char avg_pool_size[3];
x(GIR)	char cache_hit_rate[3];
x(GIR)	char cache_wait_buffer[3];
-	char unproc_atacs[10];
-	char unproc_prints[10];
-	char wait_dputs[10];
x(GIR)	char abterm_services[10];
-	char wait_resources[4];
x(GIR)	char deadlocks[10];
x(GIR)	char periodic_writes[10];
x(GIR)	char pages_pwrite[10];
x(GIR)	char logfile_writes[10];
-	char curr_jr[3];
x(GIR)	char maximum_jr[3];
-	char program_fgg[4];
-	char uslog_fgg[4];
x(GIR)	char max_mpgpool_size[3]; [2]
-	char curr_mpgpool_size[3]; [2]

mod [1]	Datenstruktur kc_curr_par_str
x(GIR)	`char avg_mpgpool_size[3];` [2]
x(GIR)	`char max_load[3];`
-	`char curr_load[3];`
x(GIR)	`char max_wait_resources[4];`
-	`char wait_system_resources[4];`
x(GIR)	`char max_wait_system_resources[4];`
x(GIR)	`char nr_cache_rqs[10];`
x(GIR)	`char nr_cache_searches[10];`
-	`char nr_res_rqs[10];`
x(GIR)	`char nr_res_rqs_for_max[10];`
-	`char nr_sys_res_rqs[10];`
x(GIR)	`char nr_sys_res_rqs_for_max[10];`
-	`char curr_system_tasks[3];`
x(GID)	`char data_compression;`
x(GIR)	`char avg_saved_pgs_by_compr[3];`
-	`char gen_date_year[4];`
-	`char gen_date_month[2];`
-	`char gen_date_day[2];`
-	`char gen_time_hour[2];`
-	`char gen_time_min[2];`
-	`char gen_time_sec[2];`

1 Feldinhalt mit KC_MODIFY_OBJECT modifizierbar; siehe Abschnitt "obj_type = KC_CURR_PAR"

2 UTM-internes Feld; der Feldinhalt ist irrelevant und wird im Folgenden nicht beschrieben.

Die Felder der Datenstruktur haben die folgende Bedeutung:

appliname

Name der UTM-Anwendung, der bei der KDCDEF-Generierung in MAX APPLINAME festgelegt wurde.

appliname ist der Name der Anwendung, der beim Verbindungsaufbau vom Terminal angegeben werden muss.

utm_version

Eingesetzte openUTM-Version einschließlich Korrekturstand, z.B. V07.0A00.

applimode

'S' Die Anwendung ist als UTM-S-Anwendung generiert (Secure).

'F' Die Anwendung ist als UTM-F-Anwendung generiert (Fast).

start_date_year, start_date_month, start_date_day

UTM-S-Anwendung: Datum des letzten Kaltstarts der Anwendung UTM-F-Anwendung: Datum des letzten Starts der Anwendung

start_time_hour, start_time_min, start_time_sec

UTM-S-Anwendung: Uhrzeit des letzten Kaltstarts der Anwendung UTM-F-Anwendung: Uhrzeit des letzten Starts der Anwendung

curr_date_year, curr_date_month, curr_date_day

aktuelles Datum

curr_time_hour, curr_time_min, curr_time_sec

aktuelle Uhrzeit

term_input_msgs

Anzahl aller Nachrichten, die die Anwendung seit dem letzten Zurücksetzen des Zählers *term_input_msgs* von Clients oder Partner-Anwendungen empfangen hat.

UTM setzt den Zähler automatisch auf 0 zurück bei jedem Anwendungsstart und jede volle Stunde, wenn bei der KDCDEF-Generierung MAX STATISTICS-MSG=FULL-HOUR (Standardwert) gesetzt wurde.

term_input_msgs können Sie auf 0 zurücksetzen.

term_output_msgs

Anzahl aller Nachrichten, die die Anwendung seit dem letzten Zurücksetzen des Zählers *term_output_msgs* an Clients, Drucker oder Partner-Anwendungen gesendet hat.

UTM setzt den Zähler automatisch auf 0 zurück bei jedem Anwendungsstart und jede volle Stunde, wenn bei der KDCDEF-Generierung MAX STATISTICS-MSG=FULL-HOUR (Standardwert) gesetzt wurde.

term_output_msgs können Sie auf 0 zurücksetzen.

curr_max_asyntasks

momentan eingestellte Anzahl der Prozesse, die maximal für die Asynchron-Verarbeitung verwendet werden dürfen. *curr_max_asyntasks* wird von UTM dynamisch angepasst, wenn die Gesamtzahl der Prozesse der Anwendung oder die Maximalzahl der Prozesse für Asynchron-Verarbeitung (*kc_tasks_par_str.*

mod_max_asyntasks im Abschnitt "kc_tasks_par_str - Anzahl der Prozesse") durch die Administration geändert wird.

curr_max_tasks_in_pgwt

momentan eingestellte Anzahl der Prozesse, die maximal gleichzeitig Aufträge an Transaktionscodes bearbeiten dürfen, für die blockierende Aufrufe wie z.B. der KDCS-Aufruf PGWT (Program Wait) erlaubt sind. *curr_max_tasks_in_pgwt* wird von UTM dynamisch angepasst, wenn die Gesamtzahl der Prozesse der Anwendung oder die Prozesszahl *kc_tasks_par.mod_max_tasks_in_pgwt* (siehe Abschnitt "kc_tasks_par_str - Anzahl der Prozesse") geändert wird.

curr_tasks

enthält die Anzahl der momentan laufenden Prozesse der Anwendung.

curr_asyntasks

enthält die Anzahl der Prozesse, die momentan Asynchron-Aufträge bearbeiten.

curr_tasks_in_pgwt

enthält die Anzahl der Prozesse, die momentan Aufträge an Transaktionscodes bearbeiten, für die blockierende Funktionsaufrufe (z.B. PGWT) erlaubt sind.

tasks_waiting_in_pgwt

Aktuelle Anzahl der Prozesse, die sich aufgrund blockierender Funktionsaufrufe (z.B. KDCS-Aufruf PGWT) im Wartezustand befinden.

connected_users

Anzahl der Benutzer, die derzeit mit der Anwendung verbunden sind.

open_dial_services

Anzahl der derzeit offenen Dialog-Vorgänge.
In einer UTM-Cluster-Anwendung wird ein offener Dialog-Vorgang, der Cluster-weit gültig ist, nur gezählt, wenn der Benutzer angemeldet ist.

open_asyn_services

Anzahl der derzeit offenen Asynchron-Vorgänge.

dial_ta_per_100sec

Anzahl der im letzten abgeschlossenen 100-Sekunden-Intervall ausgeführten Dialog-Transaktionen.

asyn_ta_per_100sec

Anzahl der im letzten abgeschlossenen 100-Sekunden-Intervall ausgeführten Asynchron-Transaktionen.

dial_step_per_100sec

Anzahl der im letzten abgeschlossenen 100-Sekunden-Intervall ausgeführten Dialog-Schritte.

max_dial_ta_per_100sec

Maximale Anzahl der Dialog-Transaktionen, die innerhalb eines Zeitintervalls von 100 Sekunden ausgeführt wurden. Der angegebene Wert bezieht sich auf den aktuellen Anwendungslauf. Der Wert kann mit KC_MODIFY_OBJECT zurückgesetzt werden (siehe Abschnitt "obj_type = KC_CURR_PAR").

max_asyn_ta_per_100sec

Maximale Anzahl der Asynchron-Transaktionen, die innerhalb eines Zeitintervalls von 100 Sekunden ausgeführt wurden. Der angegebene Wert bezieht sich auf den aktuellen Anwendungslauf. Der Wert kann mit KC_MODIFY_OBJECT zurückgesetzt werden (siehe Abschnitt "obj_type = KC_CURR_PAR").

max_dial_step_per_100sec

Maximale Anzahl der Dialog-Schritte, die innerhalb eines Zeitintervalls von 100 Sekunden ausgeführt wurden. Der angegebene Wert bezieht sich auf den aktuellen Anwendungslauf. Der Wert kann mit KC_MODIFY_OBJECT zurückgesetzt werden (siehe im Abschnitt "obj_type = KC_CURR_PAR").

max_pool_size

Maximale Belegung des Pagepools in Prozent. Der Wert bezieht sich in UTM-S-Anwendungen auf die letzte KDCDEF-Generierung und in UTM-F-Anwendungen auf den aktuellen Anwendungslauf. Der Wert kann mit KC_MODIFY_OBJECT zurückgesetzt werden (siehe Abschnitt "obj_type = KC_CURR_PAR").

curr_pool_size

Aktuelle Belegung des Pagepools in Prozent.

avg_pool_size

Mittlere Belegung des Pagepools in Prozent. Der Wert bezieht sich in UTM-S-Anwendungen auf die letzte KDCDEF-Generierung und in UTM-F-Anwendungen auf den aktuellen Anwendungslauf. Der Wert kann mit KC_MODIFY_OBJECT zurückgesetzt werden (siehe Abschnitt "obj_type = KC_CURR_PAR").

cache_hit_rate

Trefferquote bei der Suche einer Seite im Cache-Speicher. Die Angabe erfolgt in Prozent. Der angegebene Wert bezieht sich auf den aktuellen Anwendungslauf. Der Wert kann mit KC_MODIFY_OBJECT zurückgesetzt werden (siehe Abschnitt "obj_type = KC_CURR_PAR"). Bei einem Zurücksetzen dieses Werts werden implizit auch die Werte *cache_wait_-buffer*, *nr_cache_rqs* und *nr_cache_searches* auf 0 zurückgesetzt.

cache_wait_buffer

Prozentsatz der Anforderungen von Puffern im Cache, die zu einer Wartezeit geführt haben. *cache_wait_buffer* berücksichtigt alle Pufferanforderungen seit dem letzten Zurücksetzen des Zählers.

UTM setzt den Zähler automatisch auf 0 zurück bei jedem Anwendungsstart und jede volle Stunde, wenn bei der KDCDEF-Generierung MAX STATISTICS-MSG=FULL-HOUR (Standardwert) gesetzt wurde.

Sie können den Zähler mit KC_MODIFY_OBJECT zurücksetzen (siehe Abschnitt "obj_type = KC_CURR_PAR"). Bei einem Zurücksetzen dieses Werts werden implizit auch die Werte *cache_hit_rate*, *nr_cache_rqs* und *nr_cache_searches* auf 0 zurückgesetzt.

unproc_atacs

Anzahl der Hintergrundaufträge, die derzeit in UTM gespeichert und noch nicht vollständig bearbeitet sind. Das entspricht der Anzahl der derzeit in allen Message Queues von Asynchron-Vorgängen zwischengespeicherten Nachrichten.

unproc_prints

Anzahl der derzeit in den Message Queues aller Drucker zwischengespeicherten Nachrichten.

wait_dputs

Anzahl der derzeit wartenden zeitgesteuerten Aufträge (DPUTs).

abterm_services

Anzahl der abnormal beendeten Vorgänge seit dem letzten Zurücksetzen des Wertes. Sie können *abterm_services* mit KC_MODIFY_OBJECT zurücksetzen.

wait_resources

Dieser Wert gibt die mittlere Lockkonfliktrate der Speicherbereiche GSSB, ULS und TLS im letzten abgeschlossenen 100-Sekunden-Intervall in der Einheit Promille an, d.h. die Anzahl Wartesituationen bei Lockanforderungen pro Anzahl der Lockanforderungen für GSSB, ULS und TLS insgesamt im letzten abgeschlossenen 100-Sekunden-Intervall multipliziert mit 1000.

Ein hoher Wert in *wait_resources* kann folgende Ursachen haben:

- Prozesse mit zu langen Lauf- oder Wartezeiten,
- Betriebsmittel sind zu lange gesperrt, z.B. häufige PEND KP- oder PGWT-Aufrufe in KDCS-Teilprogrammen.

> **i** Geht ein Lock-Halter in den Status PEND KP, so werden alle "Waiter" benachrichtigt und alle weiteren Sperren sofort abgewiesen. D.h. der Wert von *wait_resources* erhöht sich dadurch nicht.

deadlocks

nzahl der erkannten und aufgelösten Deadlocks von UTM-Betriebsmitteln seit dem letzten Zurücksetzen des Wertes.

Sie können *deadlocks* mit KC_MODIFY_OBJECT zurücksetzen.

periodic_writes

Anzahl der Periodic Writes seit dem letzten Start der Anwendung bzw. seit dem letzten Zurücksetzen des Wertes mit KC_MODIFY_OBJECT. (periodic write = Sicherung der gesamten sicherungsrelevanten Verwaltungsdaten der UTM-Anwendung.)

pages_pwrite

Anzahl der UTM-Seiten, die bei einem Periodic Write im Mittel gesichert wurden.Es werden alle Periodic Writes seit dem letzten Zurücksetzen des Wertes berücksichtigt. Sie können den Wert mit KC_MODIFY_OBJECT auf 0 zurücksetzen. UTM setzt *pages_pwrite* bei jedem Anwendungsstart automatisch auf 0 zurück.

logfile_writes

Anzahl der Anforderungen, Protokollsätze auf die Benutzer-Protokolldatei (USLOG) zu schreiben, seit dem letzten Zurücksetzen des Wertes.
UTM setzt den Zähler automatisch auf 0 zurück bei jedem Anwendungsstart und jede volle Stunde, wenn bei der KDCDEF-Generierung MAX STATISTICS-MSG=FULL-HOUR (Standardwert) gesetzt wurde.

Sie können den Zähler mit KC_MODIFY_OBJECT zurücksetzen (siehe Abschnitt "obj_type = KC_CURR_PAR").

curr_jr Nur bei verteilter Verarbeitung:

Aktuelle Anzahl der gleichzeitig adressierten Auftragnehmer-Vorgänge relativ zum Generierungswert MAXJR, Angabe in Prozent.

(MAXJR = maximal Anzahl ferner Auftragnehmer-Vorgänge, die gleichzeitig in der lokalen Anwendung adressiert sein dürfen; siehe *kc_utmd_par_str* im Abschnitt "kc_utmd_par_str - Parameter für die verteilte Verarbeitung".)

maximum_jr

Nur bei verteilter Verarbeitung:

Maximale Anzahl der in der lokalen Anwendung gleichzeitig adressierten fernen Auftragnehmer-Vorgänge relativ zum Generierungswert MAXJR (siehe *kc_utmd_par_str* im Abschnitt "kc_utmd_par_str - Parameter für die verteilte Verarbeitung"). Die Angabe erfolgt in Prozent.

maximum_jr berücksichtigt alle Anforderungen an ferne Auftragnehmer-Vorgänge seit dem letzten Zurücksetzen des Wertes. *maximum_jr* können Sie mit KC_MODIFY_OBJECT auf 0 zurücksetzen.

program_fgg

Auf BS2000-Systemen: 0

Auf Unix-, Linux- und Windows-Systemen: Nummer der aktuell geladenen Dateigeneration des Anwendungsprogramms.

uslog_fgg

Nummer der Dateigeneration der Benutzer-Protokolldatei (USLOG), in die aktuell geschrieben wird.

max_load

zeigt die maximale Auslastung der UTM-Anwendung in Prozent, die seit Anwendungsstart oder dem letzten Zurücksetzen registriert wurde.

Der Wert in *max_load* kann auf den Wert in *curr_load* zurückgesetzt werden.

curr_load

zeigt die momentane Auslastung der UTM-Anwendung in Prozent, die während des letzten abgeschlossenen 100-Sekunden-Intervalls registriert wurde.

max_wait_resources

Maximale Konfliktrate für Locks auf Anwenderdaten über den Anwendungslauf. Der Wert wird in Promille angegeben.

Sie können diesen Wert mit KC_MODIFY_OBJECT zurücksetzen. Bei einem Zurücksetzen dieses Werts werden implizit auch die Werte *max_wait_system_re-sources*, *nr_res_rqs_for_max* und *nr_sys_res_rqs_for_max* auf 0 zurückgesetzt.

wait_system_resources

Mittlere Konfliktrate im letzten abgeschlossenen 100-Sekunden-Intervall für die in diesem Intervall am höchsten belastete System-Ressource. Die Ausgabe kann sich in unterschiedlichen Intervallen auf unterschiedliche System-Ressourcen beziehen. Der Wert wird in Promille angegeben.

max_wait_system_resources

Maximale Konfliktrate für Anforderungen an System-Ressourcen (Systemlocks) über den Anwendungslauf. Der Wert wird in Promille angegeben.

Sie können diesen Wert mit KC_MODIFY_OBJECT zurücksetzen. Bei einem Zurücksetzen dieses Werts werden implizit auch die Werte *max_wait_resources nr_res_rqs_for_max* und *nr_sys_res_rqs_for_max* auf 0 zurückgesetzt.

nr_cache_rqs

Anzahl von Pufferanforderungen, die für die Berechnung des Werts *cache_wait_buffer* berücksichtigt wurden.

Sie können den Wert mit KC_MODIFY_OBJECT zurücksetzen. Bei einem Zurücksetzen dieses Werts werden implizit auch die Werte *cache_hit_rate*, *cache_wait_buffer* und *nr_cache_searches* auf 0 zurückgesetzt.

nr_cache_searches

Anzahl der Suchvorgänge nach UTM-Seiten im Cache, die für die Berechnung des Wertes *cache_hit_rate* berücksichtigt wurden.

Sie können den Wert mit KC_MODIFY_OBJECT zurücksetzen. Bei einem Zurücksetzen dieses Werts werden implizit auch die Werte *cache_hit_rate*, *cache_wait_buffer* und *nr_cache_rqs* auf 0 zurückgesetzt.

nr_res_rqs

Anzahl der Anforderungen an Transaktions-Ressourcen im letzten abgeschlossenen 100-Sekunden-Intervall, die für die Berechnung des Wertes *wait_resources* berücksichtigt wurden.

nr_res_rqs_for_max

Anzahl der Anforderungen an Transaktions-Ressourcen in dem 100 Sekunden Intervall, in dem die maximale Konfliktrate *max_wait_resources* erreicht wurde.

Sie können den Wert mit KC_MODIFY_OBJECT zurücksetzen. Bei einem Zurücksetzen dieses Werts werden implizit auch die Werte *max_wait_resources*, *max_wait_system_resources* und *nr_sys_res_rqs_for_max* auf 0 zurückgesetzt.

> **i** Die Werte *nr_res_rqs* und *nr_res_rqs_for_max* sind hilfreich, um die Relevanz einer hohen Lockkonfliktrate, insbesondere die Verluste durch Lockkonflikte beurteilen zu können.

Beispiel:

nr_res_rqs=100, *wait_resources*=5
nr_res_rqs_for_max=10, *max_wait_resources*=50.

D.h. die maximale Lockkonfliktrate von 50 Prozent wurde bei 10 angeforderten Locks in 100 Sekunden erreicht, wobei auf 5 Locks wegen Konflikt gewartet werden musste.

Außerdem wurde die aktuelle Lock-Konfliktrate von 5 Prozent bei 100 angeforderten Locks in 100 Sekunden erreicht, wobei wieder auf 5 Locks gewartet werden musste.

nr_sys_res_rqs

Anzahl der Anforderungen an System-Ressourcen im letzten abgeschlossenen 100-Sekunden-Intervall, die für die Berechnung des Wertes *wait_system_resources* berücksichtigt wurden.

nr_sys_res_rqs_for_max

Anzahl der Anforderungen an System-Ressourcen in dem 100-Sekunden-Intervall, in dem die maximale Konfliktrate *max_wait_system_resources* erreicht wurde.

Sie können den Wert mit KC_MODIFY_OBJECT zurücksetzen. Bei einem Zurücksetzen dieses Werts werden implizit auch die Werte *max_wait_resources*, *max_wait_system_resources* und *nr_res_rqs_for_max* auf 0 zurückgesetzt.

curr_system_tasks

Anzahl der momentan laufenden UTM-System-Prozesse.

data_compression

gibt an, ob die Datenkomprimierung aktuell eingeschaltet ist:

'Y' die Datenkomprimierung ist eingeschaltet.

'N' die Datenkomprimierung nicht eingeschaltet.

Sie können den Wert mit KC_MODIFY_OBJECT ändern, falls Datenkomprimierung per Generierung erlaubt ist (siehe Abschnitt „kc_max_par_str - Maximalwerte der Anwendung (MAX-Parameter)" und openUTM-Handbuch „Anwendungen generieren“, MAX DATA-COMPRESSION=).

Eine Änderung wirkt über den Anwendungslauf hinaus, in UTM-Cluster-Anwendungen wirkt sie auf alle Knoten-Anwendungen.

avg_saved_pgs_by_compr

Durchschnittswert der pro Datenkomprimierung eingesparten UTM-Seiten. Nicht berücksichtigt in diesem Statistikwert wird das Schreiben von Bereichen, bei denen UTM keine Komprimierung durchführt, weil z.B. die Datenlänge kleiner als eine UTM-Seite ist. Es werden zwei Vorkomma- und eine Nachkommastelle des Statistikwertes ausgegeben, d.h. ein Inhalt von 010 entspricht der durchschnittlichen Einsparung von 1,0 UTM-Seiten.

Der Wert kann mit KC_MODIFY_OBJECT zurückgesetzt werden.

Falls für die Anwendung keine Statistikwerte zur Datenkomprimierung vorliegen, dann wird binär Null ausgegeben. Dies ist in folgenden Situationen möglich:

- Die Datenkomprimierung ist ausgeschaltet
- Der Wert wurde mit KC_MODIFY_OBJECT zurückgesetzt
- Es wurde keine Datenkomprimierung durchgeführt, weil die Anwendung "kleine" Datenbereiche verwendet, bei denen eine Komprimierung nicht sinnvoll eingesetzt werden kann.

> **i** Ist der bei *avg_saved_pgs_by_compr* ausgegebene Wert kleiner als 5 - das entspricht 0,5 eingesparten UTM-Seiten pro Komprimierungsversuch -, dann sollte die Datenkomprimierung für diese Anwendung aus Performance-Gründen ausgeschaltet werden.

gen_date_yeargen_date_month
gen_date_month
gen_date_day

Datum des Generierungslaufs für die Anwendung.

gen_time_hour
gen_time_min
gen_time_sec

Uhrzeit des Generierungslaufs für die Anwendung.

11.3.2.4 kc_diag_and_account_par_str - Diagnose- und Accounting-Parameter

Für den Parametertyp KC_DIAG_AND_ACCOUNT_PAR ist die Datenstruktur *kc_diag_and_account_par_str* definiert. *kc_diag_and_account_par_str* enthält die Datenstruktur *kc_dump_event_str*, die ihrerseits *kc_insert_str* enthält.

In *kc_diag_and_account_par_str* liefert UTM bei KC_GET_OBJECT folgende Informationen zurück:

- welche Diagnosefunktionen zur Zeit eingeschaltet sind,
- ob zur Zeit das UTM-Accounting eingeschaltet ist.

Mit KC_MODIFY_OBJECT und Parametertyp KC_DIAG_AND_ACCOUNT_PAR können Sie die verschiedenen Diagnosefunktionen, den UTM-Messmonitor KDCMON und das UTM-Accounting ein- und ausschalten.

mod [1]	**Datenstruktur kc_diag_and_account_par_str**
x(GIR)	`char account;`
x(GIR)	`char calc;`
x(IR)	`char kdcmon;`
-	`char dump_msg_id[4];`
x(GIR)	`char testmode;`
x(GIR)	`char bcam_trace;`
x(GIR)	`char osi_trace;`
x(GIR)	`char osi_trace_records[5];`
x(GA)	`char sysprot_switch;`
x(GIR)	`struct kc_dump_event_str dump_event[3];`
x(IR)	`char stxit_log;` (nur auf BS2000-Systemen)
x(IR)	`char xa_debug;`
x(IR)	`char xa_debug_out;`
-	`curr_max_btrace_lth[5];`
x(IR)	`char admi_trace;`
x(IR)	`char cpic_trace;`
x(IR)	`char tx_trace;`
x(IR)	`char xatmi_trace;`

mod [1]	Datenstruktur kc_dump_event_str
x(GIR)	`char event_type[4];`
x(GIR)	`char event[4];`
x(GIR)	`struct kc_insert_str insert[3];`

mod [1]	Datenstruktur kc_insert_str
x(GIR)	`char insert_index[2];`
x(GIR)	`union kc_value value;`
x(GIR)	`char value_type;`
x(GIR)	`char comp[2];`

1 Feldinhalt mit KC_MODIFY_OBJECT modifizierbar; siehe Abschnitt "obj_type = KC_DIAG_AND_ACCOUNT_PAR"

Die Felder der Datenstruktur *kc_diag_and_account_par_str* haben die folgende Bedeutung:

account

gibt an, ob die Abrechnungsphase des Accounting eingeschaltet ist.

'Y' Die Abrechnungsphase ist eingeschaltet (ON).

'N' Die Abrechnungsphase ist ausgeschaltet (OFF).

Die Abrechnungsphase kann während des Anwendungslaufs ein- und ausgeschaltet werden.

Zum UTM-Accounting siehe auch das openUTM-Handbuch „Anwendungen generieren“ und openUTM-Handbuch „Einsatz von UTM-Anwendungen“.

calc gibt an, ob die Kalkulationsphase für das UTM-Accounting ein- oder ausgeschaltet ist.

'Y' Die Kalkulationsphase ist eingeschaltet (ON).

'N' Die Kalkulationsphase ist ausgeschaltet (OFF).

Die Kalkulationsphase kann während des Anwendungslaufs ein- bzw. ausgeschaltet werden.

kdcmon

gibt an, ob der UTM-Messmonitor KDCMON eingeschaltet ist.

'Y' KDCMON ist eingeschaltet (ON).

Die Messwerte von KDCMON können Sie mit dem UTM-Tool KDCEVAL auswerten. Wie Sie die Messdaten des Messmonitors auswerten, ist im jeweiligen openUTM-Handbuch „Einsatz von UTM-Anwendungen“ beschrieben.

'N' KDCMON ist ausgeschaltet (OFF).

KDCMON kann während des Anwendungslaufs ein- bzw. ausgeschaltet werden. Näheres zum Einsatz von KDCMON finden Sie im jeweiligen openUTM-Handbuch „Einsatz von UTM-Anwendungen“.

dump_msg_id

Der Parameter wird nicht mehr unterstützt, bleibt aber in der Struktur als Platzhalter stehen. Verwenden Sie die Datenstruktur kc_dump_event_str).

testmode

gibt an, ob der Testmodus eingeschaltet ist.
Testmodus bedeutet, dass zusätzliche UTM-interne Routinen zur Plausibilitätsprüfung ablaufen und interne TRACE-Informationen aufgezeichnet werden.

'Y' Testmodus ist eingeschaltet (ON).

'N' Testmodus ist ausgeschaltet (OFF).

Der Testmodus kann während des Anwendungslaufs ein- und ausgeschaltet werden. Aus Performance-Gründen sollte der Testmodus nur auf Anforderung des Systemdienstes eingeschaltet werden, um Diagnoseunterlagen zu erstellen.

bcam_trace

gibt an, ob der BCAM-Trace eingeschaltet ist. BCAM-Trace wird die Tracefunktion genannt, die alle Verbindungs-spezifischen Aktivitäten innerhalb einer UTM-Anwendung verfolgt (z.B. BCAM-Tracefunktion auf BS2000-Systemen).

'Y' Der BCAM-Trace ist eingeschaltet (ON).

'S' Der BCAM-Trace ist explizit für einige LTERM- oder LPAP-Partner, MUX-Partner (BS2000-Systeme) oder USER eingeschaltet (SELECT).
Es werden nur die Aktivitäten auf den Verbindungen zu den explizit angegebenen LTERM- oder LPAP-Partnern oder MUX-Partnern bzw. Benutzerkennungen protokolliert.

'N' Der BCAM-Trace ist ausgeschaltet (OFF).

Sie können die BCAM-Tracefunktion während des Anwendungslaufs ein- bzw. ausschalten.

osi_trace

gibt an, ob die OSI-Tracefunktion eingeschaltet ist.
Der OSI-Trace wird zur Diagnose bei Problemen mit OSI TP-Verbindungen der Anwendung benötigt.

'Y' Die OSI-Tracefunktion ist eingeschaltet (ON). Es werden alle Record-Typen protokolliert.

'S' Die OSI-Tracefunktion ist für bestimmte Record-Typen eingeschaltet (SELECT). Welche Record-Typen protokolliert werden und welche nicht, wird im Feld *osi_trace_records* angegeben.

'N' Die OSI-Tracefunktion ist ausgeschaltet (OFF).

osi_trace ist nur relevant, wenn in der Anwendung Objekte für die verteilte Verarbeitung über OSI TP generiert sind.

Sie können die OSI-Tracefunktion während des Anwendungslaufs ein- bzw. ausschalten. Aus Performance-Gründen sollte der OSI-Trace nur auf Anforderung des Systemdienstes eingeschaltet werden, um Diagnoseunterlagen zu erstellen.

osi_trace_records

gibt an, welche Record-Typen der OSI-Trace protokolliert.
Jedes Feldelement von *osi_trace_records* repräsentiert einen Record-Typ:

1. Feld den Record-Typ „SPI“,
2. Feld den Record-Typ „INT“
3. Feld den Record-Typ „OSS“
4. Feld den Record-Typ „SERV“
5. Feld den Record-Typ „PROT“

Die Angabe in den einzelnen Feldelementen hat folgende Bedeutung:

'Y' Die Trace-Records des dem Feldelement entsprechenden Record-Typs werden mitprotokolliert.

'N' Die Trace-Records des dem Feldelement entsprechenden Record-Typs werden nicht mitprotokolliert.

Die Record-Typen haben folgende Bedeutung:

SPI Ereignisse am XAP-TP System Programming Interface

INT Interner Ablauf im XAP-TP Baustein

OSS Ereignisse bei der Bearbeitung von OSS-Aufrufen (OSI Session Service)

SERV OSS-interne Trace-Records vom Typ O_TR_SERV

PROT OSS-interne Trace-Records vom Typ O_TR_PROT

Sie können den OSI-Trace während des Anwendungslaufs für bestimmte Record-Typen einschalten.

Das Ausschalten der Protokollierung für einzelne Record-Typen ist nicht möglich. Sie können jedoch über den Parameter *osi_trace='N'* alle Record-Typen ausschalten und danach einzelne Record-Typen wieder einschalten.

Der Inhalt von *osi_trace_records* ist nur relevant, wenn in der Anwendung Objekte für die verteilte Verarbeitung über OSI TP generiert sind.

sysprot_switch

gibt an, ob die Protokolldateien der UTM-Anwendung umgeschaltet werden sollen.

'Y' Die Protokolldateien sollen umgeschaltet werden.

'N' Die Protokolldateien sollen nicht umgeschaltet werden.

stxit_log (nur auf BS2000-Systemen)

gibt an, ob das STXIT-Logging ein- oder ausgeschaltet werden soll.

'Y' Das STXIT-Logging ist eingeschaltet.

'N' Das STXIT-Logging ist ausgeschaltet.

Sie können das STXIT-Logging während des Anwendungslaufs ein- bzw. ausschalten.

xa_debug

gibt an, ob Debug-Informationen für den XA-Anschluss an die Datenbank ausgegeben werden sollen.

'Y' XA-DEBUG ist eingeschaltet.
Die Aufrufe an der XA-Schnittstelle werden protokolliert.

'A' Erweitertes XA-DEBUG ist eingeschaltet (ALL).
Zusätzlich zu den Aufrufen an der XA-Schnittstelle werden bestimmte Datenbereiche protokolliert.

'N' XA-DEBUG ist ausgeschaltet.

Sie können XA-DEBUG während des Anwendungslaufs ein- bzw. ausschalten.

xa_debug_out

steuert die Ausgabe-Ziele von XA-DEBUG.

'S' Ausgabe auf SYSOUT/stderr, Standardwert.

'F' Ausgabe in eine Datei.

Wenn Sie nur das Feld *xa_debug* verwenden, ohne *xa_debug_out* zu versorgen, so wird ggf. der Wert verwendet, den Sie beim Starten der UTM-Anwendung im Startparameter .RMXA DEBUG= angegeben haben (siehe openUTM-Handbuch „Einsatz von UTM-Anwendungen“). Andernfalls wird auf SYSOUT /stderr protokolliert.

curr_max_btrace_lth

gibt die maximale Länge der Daten an, die bei eingeschalteter BCAM-Tracefunktion aufgezeichnet werden. Siehe auch Startparameter BTRACE.

admi_trace

gibt an, ob die ADMI-Tracefunktion (Tracefunktion der Programmschnittstelle zur Administration KDCADMI) eingeschaltet ist.

Siehe auch Startparameter ADMI-TRACE.

'Y': Die ADMI-Tracefunktion ist eingeschaltet.

'N': Die ADMI-Tracefunktion ist ausgeschaltet.

Sie können die ADMI-Tracefunktion während des Anwendungslaufs ein- bzw. ausschalten.

cpic_trace

gibt an, ob die CPI-C-Tracefunktion (Tracefunktion der X/Open-Schnittstelle CPI-C) eingeschaltet ist. Siehe auch Startparameter CPIC-TRACE.

'T'

Die CPI-C-Tracefunktion ist mit Level TRACE eingeschaltet. Zu jedem CPI-C-Funktionsaufruf wird der Inhalt der Input- und Output-Parameter ausgegeben. Von den Datenpuffern werden nur die ersten 16 Byte ausgegeben. Die Returncodes der KDCS-Aufrufe, auf die die CPI-C-Aufrufe abgebildet werden, werden ausgegeben.

'B' Die CPI-C-Tracefunktion ist mit Level BUFFER eingeschaltet. Dieser Trace-Level umfasst den Level TRACE, die Datenpuffer werden jedoch in voller Länge protokolliert.

'D' Die CPI-C-Tracefunktion ist mit Level DUMP eingeschaltet. Dieser Trace-Level umfasst den Level TRACE, zusätzlich werden Diagnose-Informationen in die Trace-Datei geschrieben.

'A' Die CPI-C-Tracefunktion ist mit Level ALL eingeschaltet. Dieser Trace-Level beinhaltet die Level BUFFER, DUMP und TRACE.

'N' Die CPI-C-Tracefunktion ist ausgeschaltet.

Sie können die CPI-C-Tracefunktion während des Anwendungslaufs ein- bzw. ausschalten.

tx_trace

gibt an, ob die TX-Tracefunktion (Tracefunktion der X/Open-Schnittstelle TX) eingeschaltet ist. Siehe auch Startparameter TX-TRACE.

'E' Die TX-Tracefunktion ist mit Level ERROR eingeschaltet. Es werden nur Fehler protokolliert.

'I' Die TX-Tracefunktion ist mit Level INTERFACE eingeschaltet. Dieser Trace-Level umfasst den Level ERROR, zusätzlich werden alle TX-Aufrufe protokolliert.

'F' Die TX-Tracefunktion ist mit Level FULL eingeschaltet. Dieser Trace-Level umfasst den Level INTERFACE, zusätzlich werden alle KDCS-Aufrufe, auf die die TX-Aufrufe abgebildet werden, protokolliert.

'D' Die TX-Tracefunktion ist mit Level DEBUG eingeschaltet. Dieser Trace-Level umfasst den Level FULL, zusätzlich werden Diagnose-Informationen protokolliert.

'N' Die TX-Tracefunktion ist ausgeschaltet.

Sie können die TX-Tracefunktion während des Anwendungslaufs ein- bzw. ausschalten.

xatmi_trace

gibt an, ob die XATMI-Tracefunktion (Tracefunktion der X/Open-Schnittstelle XATMI) eingeschaltet ist. Siehe auch Startparameter XATMI-TRACE.

'E' Die XATMI-Tracefunktion ist mit Level ERROR eingeschaltet. Es werden nur Fehler protokolliert.

'I' Die XATMI-Tracefunktion ist mit Level INTERFACE eingeschaltet. Dieser Trace-Level umfasst den Level ERROR, zusätzlich werden alle XATMI-Aufrufe protokolliert.

'F' Die XATMI-Tracefunktion ist mit Level FULL eingeschaltet. Dieser Trace-Level umfasst den Level INTERFACE, zusätzlich werden alle KDCS-Aufrufe, auf die die XATMI-Aufrufe abgebildet werden, protokolliert.

'D' Die XATMI-Tracefunktion ist mit Level DEBUG eingeschaltet. Dieser Trace-Level umfasst den Level FULL, zusätzlich werden Diagnose-Informationen protokolliert.

'N' Die XATMI-Tracefunktion ist ausgeschaltet.

Sie können die XATMI-Tracefunktion während des Anwendungslaufs ein- bzw. ausschalten.

dump_event

In der Datenstruktur *kc_dump_event_str* wird ein Ereignis festgelegt, bei dessen Eintreten ein UTM-Dump mit einem Ereignis-abhängigen Kennzeichen erzeugt wird. Der Dump wird von dem Prozess erstellt, bei dem das Ereignis aufgetreten ist. Die Anwendung wird nicht beendet. Voraussetzung für das Erstellen eines UTM-Dumps ist der eingeschaltete Testmodus (*testmode*='Y'). :

Ausführliche Informationen finden Sie im Abschnitt „KDCDIAG - Diagnosehilfen ein- und ausschalten".

Die Datenstruktur enthält eine Meldungsnummer, einen KDCS-Returncode (KDCRCCC oder KDCRCDC) oder einen SIGNON-Statuscode. Wird eine Meldung mit dieser Meldungsnummer generiert bzw. dieser Returncode oder Statuscode zurückgegeben, wird ein entsprechender UTM-Dump erzeugt.

Beschreibung der Felder der Struktur *dump_event*:

event_type

Typ des Ereignisses, für das ein UTM-Dump gezogen werden soll:

'MSG' UTM-Meldung

'RCDC' inkompatibler KDCS-Returncode

'RCCC' kompatibler KDCS-Returncode

'SIGN' SIGNON-Statuscode

'NONE' Explizites Ausschalten eines einzelnen Ereignisses für einen Message-Dump. Hiermit können die Kommandos KDCDIAG DUMP-MESSAGE [1, 2 oder 3] wieder zurückgenommen werden (siehe Abschnitt „KDCDIAG - Diagnosehilfen ein- und ausschalten").

event Meldungsnummer, KDCS-Returncode (KDCRCCC oder KDCRCDC) oder SIGNON-Statuscode, abhängig vom *event_type*:

event_type 'MSG'

vierstellige interne Meldungsnummer, mit führendem „K“ oder „P“ , z.B. K009 oder P001.

event_type 'RCDC'

inkompatibler KDCS-Returncode: KCRCDC (4 Byte), z.B. K301

event_type 'RCCC'

dreistelliger kompatibler KDCS-Returncode, z.B. 14Z

event_type 'SIGN'

SIGNON-Statuscode: KCRSIGN1 oder KCRSIGN2 (3 Byte), z.B. U01

insert

Die Angaben in der Datenstruktur *kc_insert_str* sind nur sinnvoll bei einem *event_type* 'MSG'

Ausführliche Informationen finden Sie im Abschnitt „KDCDIAG - Diagnosehilfen ein- und ausschalten".

Beschreibung der Felder der Struktur *insert*:

insert_index

Nummer des Inserts, das geprüft werden soll, z.B. „2“ für das zweite Insert einer Meldung. Sie können maximal drei Inserts pro Meldung angeben (durch die Strukturen insert[0] bis insert[2]).

Die Reihenfolge der Inserts einer UTM-Meldung finden Sie im openUTM-Handbuch „Meldungen, Test und Diagnose“

Mögliche Werte: 1 ... 20

Für einen Message-Dump unabhängig vom Insert setzen Sie alle drei *insert_index* auf „0“.

value Wert des Inserts, gegen den geprüft werden soll.

UTM stellt die Zeichenfolge in einer Union des Typs *kc_value* dar.

union kc_value
`char x[64];`
`char c[32];`

Zulässige Werte siehe *value_type*.

value_type

value_type legt fest, wie der Inhalt des Feldes *value* zu interpretieren ist:

- 'N': numerisch
- 'C': alphanumerisch
- 'X': hexadezimal

comp legt fest, ob auf Gleichheit oder Ungleichheit geprüft werden soll. Mögliche Werte sind:

'EQ' Prüfung auf Gleichheit, Standardwert

'NE' Prüfung auf Ungleichheit

11.3.2.5 kc_dyn_par_str - Dynamisch erzeugbare Objekte

Für den Parametertyp KC_DYN_PAR ist die Datenstruktur *kc_dyn_par_str* definiert. Bei KC_GET_OBJECT liefert UTM in *kc_dyn_par_str* Informationen über dynamisch erzeugbare Objekte zurück. UTM gibt für die einzelnen Objekttypen an:

- wieviele Objekte des Objekttyps insgesamt in der Konfiguration enthalten sein können,
- wieviele Objekte des Objekttyps noch mit KC_CREATE_OBJECT dynamisch in die Konfiguration aufgenommen werden können.

Datenstruktur kc_dyn_par_str
`char lterm_total[10];`
`char lterm_free[10];`
`char pterm_total[10];`
`char pterm_free[10];`
`char program_total[10];`
`char program_free[10];`
`char tac_total[10];`
`char tac_free[10];`
`char user_total[10];`
`char user_free[10];`
`char card_total[10];` (nur auf BS2000-Systemen)
`char card_free[10];` (nur auf BS2000-Systemen)
`char kset_total[10];`
`char kset_free[10];`
`char ltac_total[10];`
`char ltac_free[10];`
`char queue_total[10];`
`char queue_free[10];`
`char con_total[10];`
`char con_free[10];`
`char lses_total[10];`
`char lses_free[10];`
`char princ_total[10];` (nur auf BS2000-Systemen)
`char princ_free[10];` (nur auf BS2000-Systemen)

Die Felder der Datenstruktur haben die folgende Bedeutung:

lterm_total

gibt an, wieviele LTERM-Partner insgesamt in die Tabelle der KDCFILE eingetragen werden können. *lterm_total* ist also die Anzahl der generierten Tabellenplätze für LTERM-Partner.

Die Anzahl setzt sich zusammen aus:

- Anzahl der statisch eingetragenen LTERM-Partner.
- Anzahl der dynamisch eingetragenen LTERM-Partner (*obj_type*=KC_LTERM).
- Anzahl der LTERM-Partner von LTERM-Pools. Die Anzahl entspricht der Summe über alle NUMBER-Operanden der TPOOL-Anweisungen, die bei der KDCDEF-Generierung angegeben wurden.
- Anzahl der reservierten Tabellenplätze, die noch frei sind, d.h. in die noch LTERM-Partner eingetragen werden können.

Auch gelöschte LTERM-Partner sind in dieser Anzahl enthalten.

lterm_free

enthält die Anzahl der LTERM-Partner, die Sie noch dynamisch in die Konfiguration aufnehmen können.

pterm_total

gibt an, wieviele Clients und Drucker insgesamt in die Tabelle der KDCFILE eingetragen werden können. *pterm_total* ist also die Anzahl der generierten Tabellenplätze für Objekte des Typs KC_PTERM.

Die Anzahl setzt sich zusammen aus:

- Anzahl der statisch eingetragenen Clients und Drucker, d.h. die Anzahl der PTERM-Anweisungen in der KDCDEF-Generierung.
- Anzahl der dynamisch eingetragenen Clients/Drucker (*obj_type*=KC_PTERM).
- Anzahl der in LTERM-Pools zusammengefassten Anschlüsse für Clients. Die Anzahl entspricht der Summe über alle NUMBER-Operanden der TPOOL-Anweisungen, die bei der KDCDEF-Generierung angegeben wurden.
- Anzahl der reservierten Tabellenplätze, die noch frei sind, d.h. in die noch Clients und Drucker eingetragen werden können.

Auch gelöschte Clients und Drucker sind in dieser Anzahl enthalten.

pterm_free

enthält die Anzahl der Clients und Drucker, die Sie noch mit KC_CREATE_OBJECT eintragen können.

program_total

gibt an, wieviele Teilprogramme insgesamt in die Tabelle der KDCFILE eingetragen werden können. *program_total* ist also die Anzahl der generierten Tabellenplätze für Objekte des Typs KC_PROGRAM.

Die Anzahl setzt sich zusammen aus:

- Anzahl der statisch eingetragenen Teilprogramme und VORGANG-Exits, d.h. die Anzahl der PROGRAM-Anweisungen in der KDCDEF-Generierung.
- Anzahl der dynamisch eingetragenen Teilprogramme und VORGANG-Exits (*obj_type* =KC_PROGRAM).
- Anzahl der reservierten Tabellenplätze, die noch frei sind.

Auch gelöschte Teilprogramme sind in dieser Anzahl enthalten.

program_free

enthält die Anzahl der Teilprogramme und VORGANG-Exits, die Sie noch mit KC_CREATE_OBJECT eintragen können.

tac_total

gibt an, wieviele Transaktionscodes und TAC-Queues insgesamt in die Tabelle der KDCFILE eingetragen werden können. *tac_total* ist also die Anzahl der generierten Tabellenplätze für Objekte des Typs KC_TAC.

Die Anzahl setzt sich zusammen aus:

- Anzahl der statisch eingetragenen Transaktionscodes und TAC-Queues, d.h. die Anzahl der TAC-Anweisungen in der KDCDEF-Generierung.
- Anzahl der dynamisch eingetragenen Transaktionscodes und TAC-Queues (*obj_type*=KC_TAC).
- Anzahl der reservierten Tabellenplätze, die noch frei sind.

Auch gelöschte Transaktionscodes und TAC-Queues sind in dieser Anzahl enthalten.

tac_free

enthält die Anzahl der Transaktionscodes und TAC-Queues, die Sie noch dynamisch mit KC_CREATE_OBJECT eintragen können.

user_total

gibt an, wieviele Benutzerkennungen insgesamt in die Tabelle der KDCFILE eingetragen werden können. *user_total* ist also die Anzahl der generierten Tabellenplätze für Objekte des Typs KC_USER.

Die Anzahl setzt sich zusammen aus:

- Anzahl der statisch und dynamisch eingetragenen Benutzerkennungen (in einer Anwendung, die mit Benutzerkennungen generiert ist) bzw.
  Anzahl der statisch oder dynamisch eingetragenen LTERM-Partner (in einer Anwendung, die ohne Benutzerkennungen generiert ist).
- Anzahl der existierenden Clients mit *ptype*='APPLI' (TS-Anwendungen, die keine Socket-Anwendungen sind), *ptype*='UPIC-...' (UPIC-Clients) oder
  ptype='SOCKET' (Socket-Anwendungen). Für diese Clients erzeugt UTM intern Benutzerkennungen mit dem Namen des zugehörigen LTERM-Partners.
- Anzahl der reservierten Tabellenplätze für Benutzerkennungen, die noch frei sind (in einer Anwendung, die mit Benutzerkennungen generiert ist)
  bzw.
  Anzahl der reservierten Tabellenplätze für LTERM-Partner, die noch frei sind (in einer Anwendung, die ohne Benutzerkennungen generiert ist).

Auch gelöschte Benutzerkennungen sind in dieser Anzahl enthalten.

user_free

enthält die Anzahl der Benutzerkennungen, die Sie noch dynamisch mit KC_CRE-ATE_OBJECT eintragen können.

card_total (nur auf BS2000-Systemen)

gibt an, wieviele Benutzerkennungen mit Ausweiskarte insgesamt in die Tabelle der KDCFILE eingetragen werden können. *card_total* setzt sich zusammen aus:

- der Anzahl statisch oder dynamisch eingetragener Benutzerkennungen mit Ausweiskarte.
- Anzahl der reservierten Tabellenplätze für Benutzerkennungen mit Ausweiskarte, die noch frei sind.

card_free (nur auf BS2000-Systemen)

enthält die Anzahl der Benutzerkennungen mit Ausweiskarte, die Sie noch mit KC_CREATE_OBJECT eintragen können.

kset_total

enthält die Anzahl der Keysets, die insgesamt in die KSET-Tabelle eingetragen werden können.

kset_free

enthält die aktuelle Anzahl der Keysets, die Sie noch mit KC_CREATE_OBJECT in die KSET-Tabelle eintragen können.

ltac_total

enthält die Anzahl der LTACs, die insgesamt in die LTAC-Tabelle eingetragen werden können.

ltac_free

enthält die aktuelle Anzahl der LTACs, die Sie noch mit KC_CREATE_OBJECT in die LTAC-Tabelle eintragen können.

queue_total

enthält die Anzahl der temporären Queues, die insgesamt in die QUEUE-Tabelle eingetragen werden können. Dieser Wert wurde mit der QUEUE-Anweisung bei der Generierung festgelegt.

queue_free

enthält die aktuelle Anzahl der temporären Queues, die Sie noch mit dem KDCS-Aufruf QCRE eintragen können.

con_total

enthält die Anzahl der LU6.1-Transportverbindungen, die insgesamt in die PTERM-Tabelle eingetragen werden können.

con_free

enthält die aktuelle Anzahl der LU6.1-Transportverbindungen, die Sie noch mit KC_CREATE_OBJECT in die PTERM-Tabelle eintragen können.

lses_total

enthält die Anzahl der LU6.1-Sessions, die insgesamt in die USER-Tabelle eingetragen werden können.

lses_free

enthält die aktuelle Anzahl der LU6.1-Sessions, die Sie noch mit KC_CREATE_-OBJECT in die USER-Tabelle eintragen können.

princ_total (nur auf BS2000-Systemen)

enthält die Anzahl der insgesamt angelegten USER mit Principal.

princ_free (nur auf BS2000-Systemen)

enthält die Anzahl der USER mit Principal, die noch erzeugt werden können.

11.3.2.6 kc_max_par_str - Maximalwerte der Anwendung (MAX-Parameter)

Für den Parametertyp KC_MAX_PAR ist die Datenstruktur *kc_max_par_str* definiert. In *kc_max_par_str* liefert UTM bei KC_GET_OBJECT folgende Informationen zurück:

- Basiseigenschaften der Anwendung, z.B. den Anwendungsnamen, Funktionsvariante, Name der KDCFILE.
- die Maximalwerte der Anwendung, z.B. Größe des Pagepools, des Wiederanlaufbereichs und der KDCS-Speicherbereiche, maximale Anzahl der Benutzer, maximale Anzahl der Lock- und Keycodes der Anwendung, maximale Zeitspanne für zeitgesteuerte Asynchron-Aufträge, maximale Anzahl der für die Anwendung einsetzbaren Prozesse.
- Nur auf Unix-, Linux- und Windows-Systemen: Betriebsmittel des Systems, die von der Anwendung belegt werden, z.B. Schlüssel für Shared Memory Segmente und Semaphore.

mod [1]	**Datenstruktur kc_max_par_str**	**siehe** [2]
-	`char adf_name[16];` [3]	
-	`char applimode;`	applimode
-	`char appliname[8];`	appliname
-	`char asyntasks[3];`	asyntasks
-	`char blksize[2];`	blksize
x(GIR)	`char bretrynr[5];`	bretrynr
-	`char cacheshmkey[10];` (nur auf Unix-, Linux- und Windows-Systemen)	cacheshmkey
-	`char cachesize_pages[10];`	cachesize_pages
x(GIR)	`char cachesize_paging[3];`	cachesize_paging
-	`char cachesize_res;` (nur auf BS2000-Systemen)	cachesize_res
-	`char cardlth[3];` (nur auf BS2000-Systemen)	cardlth
-	`char catid_a[4];` (nur auf BS2000-Systemen)	catid_a
-	`char catid_b[4];` (nur auf BS2000-Systemen)	catid_b
-	`union kc_clear_char clrch;`	clrch
-	`char clrch_type;`	clrch_type
x(IR)	`char conn_users[10];`	conn_users
x(GPD)	`char destadm[8];`	destadm
-	`char dputlimit1_day[3];`	dputlimit1_day
-	`char dputlimit1_hour[2];`	dputlimit1_hour
-	`char dputlimit1_min[2];`	dputlimit1_min
-	`char dputlimit1_sec[2];`	dputlimit1_sec
-	`char dputlimit2_day[3];`	dputlimit2_day
-	`char dputlimit2_hour[2];`	dputlimit2_hour
-	`char dputlimit2_min[2];`	dputlimit2_min
-	`char dputlimit2_sec[2];`	dputlimit2_sec

mod [1]	Datenstruktur kc_max_par_str	siehe [2]
-	`char gssbs[10];`	gssbs
-	`char hostname[8];`	hostname
-	`char ipcshmkey[10];` (nur auf Unix-, Linux- und Windows-Systemen)	ipcshmkey
-	`char ipctrace[10];` (nur auf Unix-, Linux- und Windows-Systemen)	ipctrace
-	`char kaashmkey[10];` (nur auf Unix-, Linux- und Windows-Systemen)	kaashmkey
-	`char kb[10];`	kb
-	`char kdcfile_name[42];`	kdcfile_name
-	`char kdcfile_operation;`	kdcfile_operation
-	`char keyvalue[4];`	keyvalue
-	`char locale_lang_id[2];` (nur auf BS2000-Systemen)	locale_lang_id
-	`char locale_terr_id[2];` (nur auf BS2000-Systemen)	locale_terr_id
-	`char locale_ccsname[8];` (nur auf BS2000-Systemen)	locale_ccsname
-	`char lputbuf[4];`	lputbuf
-	`char lputlth[10];`	lputlth
-	`char lssbs[4];`	lssbs
-	`char mp_wait_sec[5]`	mp_wait_sec
-	`char nb[10];`	nb
-	`char net_access;` (nur auf Unix-, Linux- und Windows-Systemen)	net_access
-	`char nrconv[2];`	nrconv
-	`char osi_scratch_area[5];`	osi_scratch_area
-	`char osishmkey[10];` (nur auf Unix-, Linux- und Windows-Systemen)	osishmkey
-	`char pgpool_pages[10];`	pgpool_pages
-	`char pgpool_warnlevel1[2];`	pgpool_warnlevel1
-	`char pgpool_warnlevel2[3];`	pgpool_warnlevel2
-	`char pgpoolfs[5];`	pgpoolfs
-	`char pisizelth[5];` (nur auf Unix- und Linux-Systemen)	pisizelth

mod [1]	Datenstruktur kc_max_par_str	siehe [2]
-	`char recbuf_pages[10];`	recbuf_pages
-	`char recbuf_lth[10];`	recbuf_lth
-	`char recbuffs[3];`	recbuffs
-	`char reqnr[3];` (nur auf BS2000-Systemen)	reqnr
-	`char seclev;` [4]	
-	`char sat;` (nur auf BS2000-Systemen)	sat
-	`char semarray_startkey[10];` (nur auf Unix-, Linux- und Windows-Systemen)	semarray_startkey
-	`char semarray_number[4];` (nur auf Unix-, Linux- und Windows-Systemen)	semarray_number
-	`char semkey[10][10];` (nur auf Unix-, Linux- und Windows-Systemen)	semkey
-	`char signon_value[3];`	signon_value
-	`char signon_restr;`	signon_restr
x (GIR)	`char signon_fail[3];`	signon_fail
x (GIR)	`char sm2;`	sm2
-	`char spab[10];`	spab
-	`char syslog_size[10];`	syslog_size
-	`char tasks[3];`	tasks
-	`char tasks_in_pgwt[3];`	tasks_in_pgwt
-	`char tracerec[5];`	tracerec
-	`char trmsglth[10];`	trmsglth
-	`char uslog;`	uslog
-	`char vgmsize[3];` (nur auf BS2000-Systemen)	vgmsize
-	`char xaptpshmkey[10];` (nur auf Unix-, Linux- und Windows-Systemen)	xaptpshmkey
-	`char mpgpool_pages[10];` [4]	
-	`char mpgpool_res;` [4]	

mod [1]	Datenstruktur kc_max_par_str	siehe [2]
-	`char rtimer;` [4]	
-	`char spin_lock_asyn[10];` [4]	
-	`char spin_lock_cache[10];` [4]	
-	`char spin_lock_kaa[10];` [4]	
-	`char spin_lock_ipc[10];` [4]	
-	`char spin_lock_pcmm[10];` [4]	
-	`char xopen_cpic_dspl[5];` [4]	
-	`char xopen_cpic_lth[5];` [4]	
-	`char xopen_xatmi_dspl[5];` [4]	
-	`char xopen_xatmi_lth[5];` [4]	
-	`char xopen_tx_dspl[5];` [4]	
-	`char xopen_tx_lth[5];` [4]	
-	`char max_statistics_msg;`	max_statistics_msg
-	`char max_open_asyn_conv[10];`	max_open_asyn_conv
-	`char dead_letter_q_alarm[10];`	dead_letter_q_alarm
-	`char max_suspended_ta[3];` [4]	
-	`char atac_redelivery[3];`	atac_redelivery
-	`char dget_redelivery[3];`	dget_redelivery
-	`char principal_lth[3];` (nur auf BS2000-Systemen)	principal_lth
-	`char privileged_lterm[8];`	privileged_lterm
-	`char cache_location;`	cache_location
-	`char data_compression;`	data_compression
-	`char hostname_long[64];`	hostname_long
-	`char move_bundle_msgs;`	move_bundle_msgs

[1] Feldinhalt mit KC_MODIFY_OBJECT modifizierbar; siehe Abschnitt "obj_type = KC_MAX_PAR"

[2] Die Bedeutung der Felder ist an der in dieser Spalte angegebenen Stelle beschrieben.

[3] Entfallene Funktion aus einer früheren UTM-Version; das Feld wird von UTM mit Leerzeichen belegt.

[4] UTM-internes Feld zur Unterstützung der Diagnose bestimmter Fehlersituationen; der Feldinhalt ist für einen Anwender irrelevant und wird deswegen im Folgenden nicht beschrieben.

Die Felder der Datenstruktur haben die folgende Bedeutung:

applimode

gibt an, ob es sich bei der UTM-Anwendung um eine UTM-S- oder UTM-F-Anwendung handelt.

'S' Die Anwendung ist als UTM-S-Anwendung generiert (Secure).

'F' Die Anwendung ist als UTM-F-Anwendung generiert (Fast).

appliname

Name der UTM-Anwendung. Dieser Name wird bei der statischen Generierung mit dem Generierungstool KDCDEF in MAX APPLINAME definiert.
appliname ist der Name der Anwendung, der von Terminals beim Verbindungsaufbau angegeben werden muss.

asyntasks

enthält die maximale Anzahl der Prozesse der Anwendung, die die Bearbeitung von Aufträgen an Asynchron-Transaktionscodes übernehmen dürfen. *asyntasks* ist der Grenzwert für die aktuelle Anzahl der Prozesse zur Bearbeitung von Asynchron-Aufträgen, die beim Start der Anwendung oder durch die Administration dynamisch eingestellt werden kann.

blksize

gibt die Größe einer UTM-Seite an. Die Größe wird bei der KDCDEF-Generierung festgelegt und beträgt 2K, 4K oder 8K. Mögliche Werte:

'2': die Größe einer UTM-Seite beträgt 2K.

'4': die Größe einer UTM-Seite beträgt 4K.

'8': die Größe einer UTM-Seite beträgt 8K.

bretrynr

Nur auf BS2000-Systemen: gibt an, wie oft UTM versuchen soll, eine Nachricht an das Transportsystem (BCAM) zu übergeben, wenn BCAM die Nachricht nicht sofort übernehmen kann. Wird diese Anzahl in *bretrynr* überschritten, dann wird die Verbindung zum Dialog-Partner abgebaut. Der Wert von *bretrynr* hat Einfluss auf die Performance Ihrer UTM-Anwendung.

Bei der Ausgabe von Asynchron-Nachrichten an einen Dialog-Partner mit *ptype*='APPLI' (TS-Anwendung, die keine Socket-Anwendungen sind) hat *bretrynr* keine Bedeutung. Wird eine solche Nachricht vom Transportsystem wegen einer temporären Engpass-Situation abgewiesen, dann gibt UTM den Prozess zunächst frei, baut die Verbindung aber nicht ab. Nach einer Wartezeit von 3 Sekunden versucht UTM erneut bis zu dreimal, die Nachricht an BCAM zu übergeben. Gelingt die Übergabe auch dann nicht, wird wiederum 3 Sekunden gewartet, bevor die nächsten 3 Versuche unternommen werden etc.

Minimalwert: '1'
Maximalwert: '32767' (theoretischer Wert)

cacheshmkey

Nur auf Unix-, Linux- und Windows-Systemen: enthält den Schlüssel für das Shared Memory Segment, in dem die Anwendungs-globalen Puffer für die Dateizugriffe liegen. *cacheshmkey* ist auf Unix- und Linux-Systemen ein globaler Parameter. *cacheshmkey* ist eine Dezimalzahl.

cachesize_pages

gibt die Größe des Cache-Speichers in Anzahl UTM-Seiten an. Die Größe einer UTM-Seite wird im Feld *blksize* zurückgeliefert. Über den Cache-Speicher werden alle Zugriffe auf den Pagepool abgewickelt, d.h. alle Ein- und Ausgaben von LSSBs, GSSBs, TLS, LPUT- und FPUT-Nachrichten, MPUT-Nachrichten, sowie einige UTM-Verwaltungsdaten. Das Schreiben auf KDCFILE erfolgt erst dann, wenn im Cache-Speicher kein Platz mehr ist oder wenn die Transaktion beendet wird.

cachesize_paging

gibt an, wieviel Prozent des Cache-Speichers bei Engpass-Situationen auf einmal auf die KDCFILE geschrieben werden, damit der Speicherplatz im Cache für andere Daten verwendet werden kann. Der Wert von *cachesize_paging* beeinflusst die Performance Ihrer UTM-Anwendung.

UTM lagert bei einem Paging mindestens 8 UTM-Seiten aus, auch wenn der Wert von *cachesize_paging* weniger UTM-Seiten entspricht.

Minimalwert: '0', d.h. es werden 8 UTM-Seiten ausgelagert
Maximalwert: '100' (%)

cachesize_res

Nur auf BS2000-Systemen: gibt an, ob der Cache-Speicher resident angelegt ist oder nicht. Der Feldinhalt ist wie folgt zu interpretieren:

'R' Der Cache-Speicher ist resident angelegt.

'N' Der Cache-Speicher ist pageable, d.h. nicht resident angelegt.

cardlth

Länge der Ausweisinformation in Byte, die UTM abspeichert, wenn ein Ausweisleser ergänzend zur Berechtigungsprüfung beim Anmelden (KDCSIGN) verwendet wird. Die Ausweisinformation kann in einem Teilprogramm mit dem KDCS-Aufruf INFO gelesen werden.

catid_a

Nur auf BS2000-Systemen: enthält die Katalogkennung (CAT-ID), die Ihrer KDCFILE mit dem Suffix A auf dem BS2000-System zugeordnet ist.

catid_b

Nur auf BS2000-Systemen: ist nur relevant, wenn Sie die KDCFILE doppelt führen .
catid_b enthält dann die Katalogkennung, die Ihrer KDCFILE mit dem Suffix B zugeordnet ist. Wird die KDCFILE nur einfach geführt, dann ist *catid_b* = *catid_a*.

clrch

(**cl**ea**r** **ch**aracter)
enthält das Zeichen, mit dem der Kommunikationsbereich (KB) und der Standard Primäre Arbeitsbereich (SPAB) der Teilprogramme am Ende eines Dialog-Schritts überschrieben werden.

Wurde bei der Generierung kein Zeichen definiert, dann ist *clrch* mit Leerzeichen belegt und *clrch_type*='N'. Die Speicherbereiche werden dann am Ende des Dialog-Schritts nicht überschrieben.

Wurde bei der KDCDEF-Generierung ein Zeichen definiert, dann enthält *clrch* dieses Zeichen. Ist das Zeichen hexadezimal, dann wird jedes Halb-Byte als ein Zeichen dargestellt.

clrch wird in Form folgender Union zurückgeliefert:

union kc_clear_char
`char x[2];`
`char c;`

Das Feld *x* ist belegt, wenn *clrch* als Hexadezimalzeichen zurückgeliefert wird.Das Feld *c* ist belegt, wenn *clrch* als alphanumerisches Zeichen zurückgeliefert wird.

Dem Feld *clrch_type* können Sie entnehmen, wie der Inhalt von *clrch* zu interpretieren ist.

clrch_type

gibt an, wie der Inhalt des Feldes *clrch* zu interpretieren ist. Es bedeutet:

'X' In *clrch* steht ein hexadezimales Zeichen.

'C' In *clrch* steht ein abdruckbares, alphanumerisches Zeichen.

'N' Es wurde kein *clrch* -Zeichen definiert.

conn_users

maximale Anzahl der Benutzer, die gleichzeitig bei der UTM-Anwendung angemeldet sein dürfen. Unter Benutzer versteht man die Anzahl der Benutzerkennungen, die gleichzeitig angemeldet sein dürfen. Ist die Anwendung ohne Benutzerkennungen generiert, dann ist durch *conn_users* die Anzahl der Clients beschränkt, die sich über LTERM-Partner an die Anwendung anschließen.

Benutzerkennungen, die mit Administrationsberechtigung generiert wurden, können sich auch dann noch an die UTM-Anwendung anmelden, wenn die maximale Anzahl der gleichzeitig arbeitenden Benutzerkennungen bereits erreicht wurde.

conn_users='0' bedeutet, dass die Anzahl der gleichzeitig aktiven Benutzer nicht beschränkt ist.

Minimalwert: '0'
Maximalwert: '500000'
Auf Unix-, Linux- und Windows-Systemen darf beim Modifizieren kein größerer Wert angegeben werden als der Wert, der in der Generierung festgelegt wurde (MAX CONN-USERS).

destadm

enthält den Empfänger, an den UTM die Ergebnisse von KDCADM-Administrationsaufrufen sendet, die asynchron verarbeitet wurden (Asynchron-Transaktionscodes von KDCADM). Der Empfänger kann ein LTERM-Partner oder ein Asynchron-TAC oder eine TAC-Queue sein.

Enthält *destadm* Leerzeichen, dann ist kein Empfänger definiert. Die Ergebnisse der Asynchron-Transaktionscodes von KDCADM gehen verloren. In diesem Fall sollten Sie z.B. mit KC_MODIFY_OBJECT einen Empfänger definieren.

dputlimit1_day
dputlimit1_hour
dputlimit1_min
dputlimit1_sec

bestimmen die obere Grenze des Zeitintervalls, in dem ein zeitgesteuerter Auftrag ausgeführt werden muss. Zeitgesteuerte Aufträge werden mit dem KDCS-Aufruf DPUT erzeugt. Ein Teilprogrammaufruf und damit auch ein DPUT-Aufruf mit absoluter Zeitangabe kann sich so weit verzögern, dass der gewünschte Ausführungszeitpunkt des DPUTs bereits überschritten wurde. Dieser Zeitpunkt, an dem ein zeitgesteuerter Auftrag ausgeführt werden soll (angegeben beim DPUT-Aufruf), darf maximal um die in *dputlimit1* angegebene Zeitspanne **nach** dem Zeitpunkt des DPUT-Aufrufs liegen. *dputlimit1* wird wie folgt angegeben:

Anzahl Tage (*dputlimit1_day*) + Anzahl Stunden (*dputlimit1_hour*) + Anzahl Minuten (*dputlimit1_min*) + Anzahl Sekunden (*dputlimit1_sec*). Es gilt also:

Ausführungszeitpunkt < DPUT-Aufrufzeitpunkt + *dputlimit1*

dputlimit2_day
dputlimit2_hour
dputlimit2_min
dputlimit2_sec

bestimmen die untere Grenze des Zeitintervalls, in dem ein zeitgesteuerter Auftrag (DPUT-Aufruf) ausgeführt werden muss. Der Zeitpunkt, an dem ein zeitgesteuerter Auftrag ausgeführt werden soll (angegeben beim DPUT-Aufruf), darf maximal um die in *dputlimit2* angegebene Zeitspanne **vor** dem Zeitpunkt des DPUT-Aufrufs liegen. *dputlimit2* wird wie folgt angegeben:

Anzahl Tage (*dputlimit2_day*) + Anzahl Stunden (*dputlimit2_hour*) + Anzahl Minuten (*dputlimit2_min*) + Anzahl Sekunden (*dputlimit2_sec*).

Es gilt also:
Ausführungszeitpunkt > DPUT-Aufrufzeitpunkt - *dputlimit2*

Liegt der angegebene Ausführungszeitpunkt zwischen der in *dputlimit2* festgelegten Grenze und dem Aufrufzeitpunkt, so wird der DPUT sofort in einen FPUT umgewandelt.

gssbs

Maximale Anzahl von GSSBs (globale Sekundärspeicherbereiche), die gleichzeitig in der Anwendung existieren können.

hostname

BS2000-Systeme:
hostname enthält den Namen des virtuellen Hosts, auf dem (aus Sicht von BCAM) die Anwendung läuft.

Unix-, Linux- und Windows-Systeme:
hostname enthält den Namen des Hosts, der als Absenderadresse angegeben wird, wenn eine Verbindung von der UTM-Anwendung aus aufgebaut wird.
Ist dieser Name länger als 8 Zeichen, dann kann der vollständige, bis zu 64 Zeichen lange Name dem Feld *hostname_long* entnommen werden. Das Feld *hostname* enthält in diesem Fall die ersten 8 Zeichen des langen Namens.

ipcshmkey

Nur auf Unix-, Linux- und Windows-Systemen: enthält den Schlüssel für das Shared Memory Segment, das zur Interprozess-Kommunikation zwischen den Workprozessen einerseits und externen Prozessen der Anwendung andererseits dient. *ipcshmkey* ist auf Unix-, Linux- und Windows-Systemen ein globaler Parameter. *ipcshmkey* ist eine Dezimalzahl.

ipctrace

Nur auf Unix-, Linux- und Windows-Systemen: enthält die Anzahl der Einträge im Trace-Bereich des IPC.

UTM schreibt Einträge in den Trace-Bereich des IPC (Shared Memory Segment für die Interprozess-Kommunikation), wenn die UTM-Anwendung im Testmodus läuft (TESTMODE=ON). Diese Einträge enthalten interne Informationen für Diagnosezwecke. Wird die in *ipctrace* enthaltene Zahl an Einträgen überschritten, dann überschreibt UTM bereits vorhandene Einträge vom ältesten an.

kaashmkey

Nur auf Unix-, Linux- und Windows-Systemen: enthält den Schlüssel für das Shared Memory Segment, in dem die Anwendungs-globalen Daten abgelegt werden.
kaashmkey ist auf Unix-, Linux- und Windows-Systemen ein globaler Parameter.
kaashmkey ist eine Dezimalzahl.

kb

enthält die Länge des Kommunikationsbereiches (KB) in Byte. KB-Kopf und KB-Rückgabebereich sind bei der Längenangabe nicht berücksichtigt.

kdcfile_name

Basisname von KDCFILE, Benutzer-Protokolldatei USLOG und System-Protokolldatei SYSLOG (siehe auch openUTM-Handbuch „Anwendungen generieren“). *kdcfile_name* muss auch beim Start der Anwendung im Startparameter FILEBASE angegeben werden.

kdcfile_operation

gibt an, ob die KDCFILE doppelt oder einfach geführt wird. Der Inhalt von *kdcfile_operation* ist wie folgt zu interpretieren:

'D' Die KDCFILE wird doppelt geführt. Wurde die KDCFILE aufgespaltet, dann werden alle KDCFILE-Dateien doppelt geführt (siehe auch openUTM- Handbuch „Anwendungen generieren“, KDCFILE).

'S' Die KDCFILE wird einfach geführt. Wurde die KDCFILE aufgespaltet, dann werden alle KDCFILE-Dateien einfach geführt.

keyvalue

enthält die Nummer des größten Keycodes in der Anwendung und damit auch die Nummer des größten Lockcodes, der für einen Transaktionscode oder einen LTERM-Partner als Zugriffschutz vergeben werden kann.

keyvalue gibt auch die maximale Anzahl der Keycodes pro Keyset an.

locale_lang_id
locale_terr_id
locale_ccsname

Nur auf BS2000-Systemen: Diese Felder enthalten die drei Komponenten des Locale, das der UTM-Anwendung zugeordnet ist. Das Locale definiert die Standard-Sprachumgebung der Anwendung. Die Standard-Sprachumgebung wird jeder Benutzerkennung (KC_USER), jedem LTERM-Partner und jedem LTERM-Pool der Anwendung als Standardeinstellung für die Sprachumgebung zugeordnet. Die Standardeinstellung ist wirksam, solange für diese Objekte kein eigenes Locale definiert wird (siehe auch openUTM-Handbuch „Anwendungen generieren“).

locale_lang_id
enthält ein bis zu 2 Zeichen langes Sprachkennzeichen.

locale_terr_id
enthält ein bis zu 2 Zeichen langes Territorialkennzeichen.

locale_ccsname
(**c**oded **c**haracter **s**et **name**)
enthält den bis zu 8 Zeichen langen Namen eines erweiterten Zeichensatzes (CCS-Name; siehe auch Benutzerhandbuch zu XHCS).

lputbuf

enthält die Größe des Puffers, in dem UTM die mit dem KDCS-Aufruf LPUT erzeugten Sätze zwischenspeichert, bevor sie in die Benutzer-Protokolldatei (USLOG) geschrieben werden. Der Puffer liegt im Pagepool.

Die von den Teilprogrammen erzeugten LPUT-Sätze werden solange in diesem Puffer zwischengespeichert, bis dieser voll ist. Erst dann kopiert UTM die Sätze in die Benutzer-Protokolldatei. Die Benutzer-Protokolldatei (USLOG) ist nur während dieses Kopier-Vorgangs geöffnet.

lputlth

enthält die maximale Länge der Benutzerdaten in einem LPUT-Satz.
Die Länge eines LPUT-Satzes ergibt sich aus:
lputlth + 84 Byte für den KB-Kopf + 12 Byte für Längenfelder.

lssbs

enthält die maximale Anzahl von LSSBs (Lokale Sekundäre Speicherbereiche), die innerhalb eines Vorgangs erzeugt werden können.

mp_wait_sec

(**m**emory **p**ool **wait**) gibt an, wieviele Sekunden ein UTM-Anwendungsprogramm maximal auf den Anschluss eines Prozesses an einen Common Memory Pool wartet.

nb

(**N**achrichten**b**ereich)
enthält die maximale Länge des Nachrichtenbereichs für KDCS-Teilprogramme.

net_access

Nur auf Unix-, Linux- und Windows-Systemen: Dieser Parameter wird nicht mehr unterstützt.

nrconv

(**n**umbe**r** of **conv**ersations) Maximalzahl der Vorgänge, die ein Benutzer gleichzeitig kellern darf. Der Wert '0' bedeutet, dass kein Vorgang gekellert werden kann.

osi_scratch_area

Größe eines UTM-internen Arbeitsbereichs, den UTM zur dynamischen Ablage von Daten benötigt, wenn mit dem OSI TP-Protokoll gearbeitet wird. Die Angabe erfolgt in KB.

In UTM-Anwendungen auf BS2000-Systemen wird dieser Arbeitsbereich während des Anwendungslaufs bei Bedarf automatisch vergrößert.

In UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen ist die Größe des internen Arbeitsbereichs während des gesamten Anwendungslaufs fest. Wenn sich im Betrieb der interne Arbeitsbereich als nicht ausreichend erweist, muss die KDCDEF-Generierung mit einem höheren Wert wiederholt werden.

osishmkey

Nur auf Unix-, Linux- und Windows-Systemen: enthält den Schlüssel für das Shared Memory Segment, das bei der Kommunikation über OSI TP von OSS verwendet wird. *osishmkey* ist auf Unix-, Linux- und Windows-Systemen ein globaler Parameter. *osishmkey* ist eine Dezimalzahl.

pgpool_pages

gibt die Größe des Pagepools in Anzahl UTM-Seiten an. Die Größe einer UTM-Seite wird im Feld *blksize* ausgegeben.

pgpool_warnlevel1
pgpool_warnlevel2

enthalten die Warnstufen, bei denen UTM vor dem Überlauf des Pagepools warnt.

pgpool_warnlevel1
gibt an, bei welcher Belegung des Pagepool UTM die erste Warnung (Meldung K041) ausgibt. *pgpool_warnlevel1* ist eine Dezimalzahl und als Prozentangabe zu interpretieren.

pgpool_warnlevel2
gibt an, bei welcher Belegung des Pagepool UTM die zweite Warnung ausgibt. Nach Überschreiten der Warnstufe 2 werden Asynchron-Aufträge abgewiesen. In diesem Fall erhält der Benutzer die Meldung K041, einem Teilprogramm wird ein entsprechender Returncode übermittelt. *pgpool_warnlevel2* ist eine Dezimalzahl und als Prozentangabe zu interpretieren.

pgpoolfs

enthält die Anzahl der Dateien, auf die der Pagepool aufgeteilt ist. Ist *pgpoolfs*='0', dann liegt der Pagepool in der Hauptdatei der KDCFILE, d.h. der Pagepool wurde nicht ausgelagert.

Bei doppelter Führung der KDCFILE besteht der Pagepool der zweiten KDCFILE ebenfalls aus *pgpoolfs* Dateien.

pisizelth

Nur auf Unix- und Linux-Systemen: Dieser Parameter wird nicht mehr unterstützt.

recbuf_pages

enthält die Größe des Wiederanlaufbereichs pro Prozess. Die Größe wird in Anzahl UTM-Seiten angegeben. Die Größe einer UTM-Seite wird im Feld *blksize* ausgegeben.

In den Wiederanlaufbereich werden die Daten geschrieben, die für den Wiederanlauf nach einem Systemfehler benötigt werden. *recbuf_pages* beeinflusst die Performance der Anwendung: Ist dieser Bereich groß, wird die laufende Anwendung weniger belastet; der Wiederanlauf nach Systemfehlern dauert jedoch länger. Ist der Bereich klein, wird die laufende Anwendung stärker belastet; der Wiederanlauf ist jedoch schneller.

recbuf_lth

enthält die Größe des Puffers in Byte, der pro Prozess der Anwendung zur Zwischenspeicherung von Wiederanlauf-Daten zur Verfügung steht. Die Daten werden für den Wiederanlauf nach Transaktions- oder Systemfehlern benötigt.

recbuffs

gibt an, auf wieviele Dateien der Wiederanlaufbereich aufgeteilt ist.

Ist *recbuffs*='0' liegt der Wiederanlaufbereich in der Hauptdatei der KDCFILE, d.h. der Wiederanlaufbereich wurde nicht ausgelagert.

Bei doppelter Führung der KDCFILE besteht der Wiederanlaufbereich der zweiten KDCFILE ebenfalls aus *recbuffs* Dateien.

reqnr

Nur auf BS2000-Systemen: enthält die maximale Anzahl der PAM-Schreib-/Leseaufträge, die in einem UTM-Prozess gleichzeitig für eine Datei abgesetzt werden können. *reqnr* enthält den bei der KDCDEF-Generierung festgelegten Wert, solange dieser kleiner ist als der Wert von *cachesize_pages*. Ist der generierte Wert größer, wird für *reqnr* der Wert von *cachesize_pages* ausgegeben.

sat

(**s**ecurity **a**udit **t**rail, nur auf BS2000-Systemen)
gibt an, ob die SAT-Protokollierung für die Anwendung eingeschaltet ist.

Die SAT-Protokollierung kann mit dem Transaktionscode KDCMSAT ein- und ausgeschaltet werden (siehe openUTM-Handbuch „Einsatz von UTM-Anwendungen auf BS2000-Systemen“, UTM-SAT-Administration).

'Y' Die SAT-Protokollierung ist eingeschaltet (ON).

'N' Die SAT-Protokollierung ist nicht eingeschaltet (OFF).
Es wird nur jeder Zugriff auf den Transaktionscode KDCMSAT (außer KDCMSAT HELP) protokolliert. Alle anderen Ereignisse werden nicht protokolliert.

semarray_startkey
semarray_number

Nur auf Unix-, Linux- und Windows-Systemen: Diese Felder geben den Bereich der Schlüssel (Keys) für die Anwendungs-globalen Semaphore an. Semaphore dienen der Prozesssynchronisation. Die Schlüssel sind auf Unix-, Linux- und Windows-Systemen globale Parameter.

semarray_startkey
enthält die Nummer des ersten Semaphoren-Schlüssels.

semarray_number
enthält die Anzahl der Schlüssel, die für die Anwendung belegt sind.

UTM belegt diese Schlüssel, indem vom ersten Schlüssel *semarray_startkey* ausgehend jeweils 1 addiert wird.

Wurde bei der KDCDEF-Generierung kein Bereich von Schlüsseln definiert, dann liefert UTM in *semarray_startkey* und *semarray_number* den Wert '0' zurück. In diesem Fall liefert UTM die Semaphore-Schlüssel im Feld *semkey* zurück.

semkey

(**sem**aphore **key,** nur auf Unix-, Linux- und Windows-Systemen): Liefert UTM in den Feldern *semarray_startkey* und *semarray_number* Werte ungleich '0' zurück, dann ist *semkey* mit '0' belegt. Sind die Felder *semarray_startkey* und *semarray_number* mit '0' belegt, dann enthält *semkey* die Schlüssel der Anwendung für die Anwendungsglobalen Semaphore (Prozesssynchronisation). Die Schlüssel sind im System globale Parameter. Die Schlüssel werden als Dezimalzahlen angegeben. Es werden maximal 10 Schlüssel zurückgeliefert. Wurden weniger als 10 Schlüssel generiert, dann wird der Rest des Feldes mit '0' versorgt.

signon_value

gibt an, für wieviel Prozent der Benutzerkennungen gleichzeitig ein Anmelde-Vorgang aktiv sein kann. UTM versucht entsprechend dieser Angabe, die notwendigen Betriebsmittel anzulegen (siehe Abschnitt „kc_signon_str - Eigenschaften des Anmeldeverfahrens").

signon_restr

gibt an, ob Einschränkungen für den Anmelde-Vorgang generiert wurden (siehe Abschnitt „kc_signon_str - Eigenschaften des Anmeldeverfahrens"):

'R' Im 1. Teil des Anmelde-Vorgangs sind Datenbank-Aufrufe und Zugriffe auf globale UTM-Speicher verboten (RESTRICTED).

'N' Im 1. Teil des Anmelde-Vorgangs sind Datenbank-Aufrufe und Zugriffe auf globale UTM-Speicher erlaubt.

signon_fail

gibt an, nach wievielen unmittelbar aufeinanderfolgenden erfolglosen Anmeldeversuchen eines Terminal-Benutzers UTM einen „stillen Alarm“ auslöst. Stiller Alarm bedeutet, dass UTM die Meldung K094 erzeugt, in die SYSLOG schreibt und ggf. an andere, für diese Meldung konfigurierte Meldungszeile ausgibt. Siehe Abschnitt „kc_signon_str - Eigenschaften des Anmeldeverfahrens".

Minimalwert: '1'
Maximalwert: '100'

sm2

gibt an, ob die UTM-Anwendung zur Überwachung der Performance Daten an openSM2 liefert.

'0' Die Performanceüberwachung mit openSM2 ist für die UTM-Anwendung generell ausgeschlossen. D.h. die UTM-Anwendung darf keine Daten an openSM2 liefern. Die Lieferung von Daten an openSM2 kann auch durch die Administration nicht eingeschaltet werden.

'N' Die UTM-Anwendung darf Daten an openSM2 liefern. Die Datenlieferung an openSM2 ist jedoch derzeit ausgeschaltet. Sie kann durch die Administration eingeschaltet werden.

'Y' Die UTM-Anwendung darf Daten an openSM2 liefern. Die Lieferung von Daten an openSM2 ist eingeschaltet. Sie kann durch die Administration ausgeschaltet werden.

spab

enthält die maximale Länge des Standard Primären Arbeitsbereiches (SPAB).

syslog_size

enthält den generierten Schwellwert für die automatische Größenüberwachung der SYSLOG-Datei durch UTM. Die automatische Größenüberwachung der SYSLOG ist nur möglich, wenn die SYSLOG als Dateigenerationsgruppe (FGG) bzw.
Dateigenerationsverzeichnis angelegt ist (siehe auch openUTM-Handbuch „Einsatz von UTM-Anwendungen “). UTM schaltet auf die nächste Dateigeneration der SYSLOG-FGG um, wenn die Größe der Dateigeneration, die aktuell beschrieben wird, den Schwellwert *syslog_size* erreicht.

syslog_size='0' bedeutet, dass UTM die Größe der SYSLOG-Datei nicht überwacht.
UTM schreibt alle Meldungen mit dem Meldungsziel SYSLOG in dieselbe Dateigeneration.

syslog_size='0' wird immer ausgegeben, wenn die SYSLOG-Datei als einfache Datei geführt wird.

Sie können die Protokollierung auf eine andere Dateigeneration umschalten, die Größenüberwachung ein- und ausschalten oder den Schwellwert ändern (siehe KC_SYSLOG im Abschnitt "KC_SYSLOG - System-Protokolldatei administrieren" und KDCSLOG im Abschnitt "KDCSLOG - SYSLOG-Datei administrieren").

tasks

Maximale Anzahl der Prozesse, die gleichzeitig für die Anwendung eingesetzt werden können. *tasks* enthält den mit KDCDEF eingestellten Maximalwert (in MAX TASKS).

Die Anzahl der Prozesse, die Aufträge der Anwendung bearbeiten können, wird bei jedem Start der Anwendung über Startparameter neu festgelegt und kann während des Anwendungslaufs an die aktuellen Erfordernisse angepasst werden (siehe KDCAPPL im Abschnitt "KDCAPPL - Eigenschaften und Grenzwerte für den Betrieb ändern" und KC_MODIFY_OBJECT im Abschnitt "obj_type=KC_TASKS_PAR"). Weder die beim Start angegebene Anzahl der Prozesse noch die durch die Administration eingestellte Anzahl darf jedoch den in *tasks* zurückgelieferten Wert überschreiten.

tasks_in_pgwt

gibt die maximale Anzahl der Prozesse an, die gleichzeitig Aufträge mit blockierenden Aufrufen wie z.B. den KDCS-Aufruf PGWT (Program Wait) bearbeiten dürfen.

Die aktuell eingestellte Prozesszahl wird bei der Informationsabfrage mit Parametertyp KC_TASKS_PAR in *kc_tasks_par_str* zurückgeliefert.

Die aktuelle Anzahl der Prozesse wird beim Start der Anwendung festgelegt und kann in Engpasssituationen durch die Administration angepasst werden (siehe KDCAPPL im Abschnitt "KDCAPPL - Eigenschaften und Grenzwerte für den Betrieb ändern" und, KC_MODIFY_OBJECT im Abschnitt "obj_type=KC_TASKS_PAR"). Weder die beim Start angegebene Anzahl noch die durch die Administration eingestellte Anzahl darf den hier ausgegebenen Wert überschreiten.

Ist *tasks_in_pgwt*='0', dann sind keine blockierenden Aufrufe erlaubt.

tracerec

(**trace rec** ords) enthält die maximale Anzahl der Einträge im TRACE-Bereich. In diesen Bereich schreibt UTM Diagnoseinformationen, falls TESTMODE=ON gesetzt ist.
Jeder Eintrag ist auf 32-Bit-Plattformen 64 Bytes lang und auf 64-Bit-Plattformen 128 Bytes lang.

trmsglth

(**tr**ansfer **me**s**s**a**g**e **l**eng**th**) enthält die maximale Länge der physischen Nachrichten, die zwischen Clients, Partner-Anwendungen oder Druckern und der UTM-Anwendung ausgetauscht werden. Bei dieser Längenangabe werden Steuerzeichen, Positionierfolgen usw. mitgerechnet. Die Angabe erfolgt in Byte.

uslog

gibt an, ob die Benutzer-Protokolldatei aus Sicherheitsgründen doppelt geführt wird.

'S' (SINGLE)
Die Benutzer-Protokolldatei wird einfach geführt.

'D' (DOUBLE)
Die Benutzer-Protokolldatei wird doppelt geführt.

Näheres zur Benutzer-Protokolldatei finden Sie im openUTM-Handbuch „Einsatz von UTM-Anwendungen“.

vgmsize

Nur auf BS2000-Systemen: enthält die Größe des Puffers für das Vorgangsgedächtnis eines SQL-Datenbanksystems. Damit wird auch der Anteil eines Benutzers am Pagepool begrenzt. *vgmsize* wird in KB angegeben.

xaptpshmkey

Nur auf Unix-, Linux- und Windows-Systemen: enthält den Schlüssel für das Shared Memory Segment, das XAPTP bei der Kommunikation über OSI TP verwendet.

xaptpshmkey ist auf Unix-, Linux- und Windows-Systemen ein globaler Parameter. *xaptpshmkey* ist eine Dezimalzahl.

max_statistics_msg

zeigt an, ob in der Anwendung stündlich die Statistikmeldung K081 erzeugt wird oder nicht (siehe openUTM-Handbuch „Meldungen, Test und Diagnose“ K081 und openUTM-Handbuch „Anwendungen generieren“ MAX STATISTICS-MSG).

'Y' die Statistikmeldung K081 wird jede Stunde erzeugt und in die SYSLOG-Datei geschrieben.

Bei der Ausgabe der Meldung werden einige Anwendungs-spezifische Statistikwerte auf Null gesetzt.

'N' die Statistikmeldung K081 wird nicht erzeugt.

Die Anwendungs-spezifischen Statistikwerte können dann bei Bedarf durch die Administration zurückgesetzt werden (siehe KC_MODIFY_OBJECT, KC_CURR_PAR im Abschnitt "obj_type = KC_CURR_PAR").

max_open_asyn_conv

enthält die maximale Anzahl der Asynchron-Vorgänge, die gleichzeitig offen sein dürfen.

dead_letter_q_alarm

steuert die Überwachung der Anzahl von Nachrichten in der Dead Letter Queue. Jedesmal, wenn der Schwellwert erreicht ist, wird die Meldung K134 ausgegeben. Bei einem Schwellwert 0 ist die Überwachung ausgeschaltet.

atac_redelivery

enthält die maximale Anzahl wiederholter Zustellungen einer Nachricht an einen Asynchron-Vorgang bei abnormaler Beendigung des Vorgangs.

dget_redelivery

enthält die maximale Anzahl wiederholter Zustellungen einer Nachricht an eine Service-gesteuerte Queue beim Rücksetzen der Transaktion.

principal_lth

Nur auf BS2000-Systemen: enthält die maximale Länge eines Kerberos-Principals in Bytes (siehe openUTM-Handbuch „Anwendungen generieren“, MAX PRINCIPAL-LTH=).

privileged_lterm

enthält den Namen des privilegierten LTERMs (siehe openUTM-Handbuch „Anwendungen generieren“, MAX PRIVILEGED-LTERM=).

cache_location

gibt den Ablageort des UTM-Cache zurück:

'P' Der UTM-Cache ist im Programmraum angelegt.

Für Unix-, Linux- und Windows-Systeme wird hier immer der Wert 'P' zurückgegeben.

'D' Auf BS2000-Systemen ist der UTM-Cache in einem oder mehreren Datenräumen angelegt (siehe openUTM-Handbuch „Anwendungen generieren“, MAX CACHESIZE=).

data_compression

gibt an, ob die Datenkomprimierung per Generierung erlaubt ist:

'Y' die Datenkomprimierung ist erlaubt.

'N' die Datenkomprimierung nicht erlaubt.

Siehe openUTM-Handbuch „Anwendungen generieren“, KDCDEF-Anweisung MAX DATA-COMPRESSION=.

hostname_long

BS2000-Systeme:
hostname_long enthält den Namen des virtuellen Hosts, auf dem (aus BCAM-Sicht) die UTM-Anwendung läuft.

Unix-, Linux- und Windows-Systeme:
hostname_long enthält den Namen des Hosts, der als Absenderadresse angegeben wird, wenn eine Verbindung von der UTM-Anwendung aus aufgebaut wird.

move_bundle_msgs

enthält den Wert, der im Parameter MOVE-BUNDLE-MSGS der MAX-Anweisung generiert ist:

'Y' wartende Asynchron-Nachrichten eines Slave-LTERMs, Slave-LPAPs oder Slave-OSI-LPAPs, für das keine Verbindung zur Partneranwendung aufgebaut werden kann, werden von UTM auf einen anderen Slave des gleichen Bündels umgehängt.

'N' wartende Asynchron-Nachrichten an einem Slave werden nicht umgehängt.

11.3.2.7 kc_msg_dest_par_str - Eigenschaften der Benutzer-spezifischen Meldungsziele

Für den Objekttyp KC_MSG_DEST_PAR ist die Datenstruktur *kc_msg_dest_all_par_str* definiert. Diese Datenstruktur enthält wiederum die vier Strukturen *user_dest_1*, *user_dest_2*, *user_dest_3* und *user_dest_4*, in denen UTM bei KC_GET_OBJECT die Informationen über die vier Benutzer-spezifischen Meldungsziele liefert.

Ist ein Meldungsziel nicht generiert, werden Leerzeichen zurückgeliefert.

Datenstruktur kc_msg_dest_all_par_str
`struct kc_msg_dest_par_str user_dest_1;`
`struct kc_msg_dest_par_str user_dest_2;`
`struct kc_msg_dest_par_str user_dest_3;`
`struct kc_msg_dest_par_str user_dest_4;`

wobei

Datenstruktur kc_msg_dest_par_str
`char md_name[8];`
`char md_type;`
`char md_format;`

Die Felder der Datenstruktur haben folgende Bedeutung:

md_name

enthält den Namen des Benutzer-spezifischen Meldungsziels.

md_type

bezeichnet den Typ des Meldungsziels in *name*. Mögliche Werte sind:

'L' für einen LTERM-Partner.

'T' für einen TAC oder eine TAC-Queue.

'U' für eine Benutzerkennung bzw. eine USER-Queue.

md_format

zeigt das Format an, in dem Meldungen an das Meldungsziel übergeben werden. Mögliche Werte sind:

'F' (FILE)
Das Format entspricht den Datenstrukturen für das MSGTAC-Programm (siehe Abschnitt „Steuerung über MSGTAC-Programm").

'P' (PRINT)
Das Format entspricht dem Ausgabeformat des UTM-Tool KDCPSYSL (siehe openUTM-Handbuch „Einsatz von UTM-Anwendungen“).

11.3.2.8 kc_pagepool_str - aktuelle Belegung des Pagepools

Für den Parametertyp KC_PAGEPOOL ist die Datenstruktur *kc_pagepool_str* definiert. In *kc_pagepool_str* liefert UTM bei KC_GET_OBJECT Informationen zur aktuellen Belegung des Pagepools zurück.

Datenstruktur kc_pagepool_str
`char total_pages[10];`
`char free_pages[10];`
`char gssb_pages[10];`
`char lssb_pages[10];`
`char tls_pages[10];`
`char uls_pages[10];`
`char dial_conv_pages[10];`
`char tacclass_pages[10];`
`char fpmm_pages[10];`
`char fput_pages[10];`
`char msgtac_pages[10];`
`char lput_pages[10];`
`char phys_msg_pages[10];`
`char reset_msg_pages[10];`
`char log_rec_pages[10];`
`char other_pages[10];`

Die Felder der Datenstruktur haben folgende Bedeutung:

total_pages

Gesamtanzahl der Seiten im Pagepool.

free_pages

Anzahl der freien Seiten.

gssb_pages

Anzahl der Seiten, die für GSSBs belegt sind.

lssb_pages

Anzahl der Seiten, die für LSSBs belegt sind.

tls_pages

Anzahl der Seiten, die für TLS-Bereiche belegt sind.

uls_pages

Anzahl der Seiten, die für ULS-Bereiche belegt sind.

dial_conv_pages

Anzahl der Seiten, die für Vorgangskontexte von Benutzern belegt sind.

tacclass_pages

Anzahl der Seiten, die für Dialogeingabenachrichten belegt sind, die temporär in TAC-Klassen Queues zwischengespeichert sind.

fpmm_pages

Anzahl der Seiten, die für die Verwaltung von Asynchronnachrichten benötigt werden.

fput_pages

Anzahl der Seiten, die für Asynchronnachrichten belegt sind.

msgtac_pages

Anzahl der Seiten, die für MSGTAC-Nachrichten belegt sind.

lput_pages

Anzahl der Seiten, die für temporär zwischengespeicherte LPUT-Sätze belegt sind.

phys_msg_pages

Anzahl der Seiten, die für Ausgabenachrichten belegt sind, die temporär zwischengespeichert werden müssen, weil sie aufgrund ihrer Länge nur abschnittsweise an das Transportsystem übergeben werden können.

reset_msg_pages

Anzahl der Seiten, die für Rücksetznachrichten belegt sind.

log_rec_pages

Anzahl der Seiten, die für OSI TP Log-Records belegt sind.

other_pages

Anzahl anderer belegter Seiten.

i Bei UTM-Cluster-Anwendungen werden GSSB- und ULS-Bereiche im globalen Pagepool der UTM-Cluster-Anwendung abgelegt. Da KC_PAGEPOOL nur die Belegung des lokalen Pagepools ausgibt, sind in UTM-Cluster-Anwendungen die Werte für *gssb_pages* und *uls_pages* immer Null.

11.3.2.9 kc_queue_par_str - Eigenschaften von Queue-Objekten

Für den Parametertyp KC_QUEUE_PAR ist die Datenstruktur *kc_queue_par_str* definiert. In *kc_queue_par_str* liefert UTM bei KC_GET_OBJECT allgemeine Informationen zu temporären Queues zurück.

Datenstruktur kc_queue_par_str
`char qp_number[10];`
`char qlev[5];`
`char qmode;`

Die Felder der Datenstruktur haben folgende Bedeutung:

qp_number

Generierte maximale Anzahl von Queue-Objekten, die während eines Anwendungslaufs zu einer Zeit existieren können.

qlev Standardwert beim Erzeugen einer temporären Queue:
Die maximale Anzahl von Nachrichten, die gleichzeitig in einer temporären Queue stehen können.

qmode

Standardwert beim Erzeugen einer temporären Queue:
Verhalten von UTM beim Überschreiten der maximal erlaubten Anzahl von Nachrichten in der Queue. Mögliche Werte sind:

'S' (STD)
UTM lehnt weitere Nachrichten für diese Queue ab.

'W' (WRAP-AROUND)
UTM nimmt weitere Nachrichten auf. Beim Eintrag einer neuen Nachricht wird die älteste in der Queue stehende Nachricht gelöscht.

11.3.2.10 kc_signon_str - Eigenschaften des Anmeldeverfahrens

Für den Objekttyp KC_SIGNON ist die Datenstruktur *kc_signon_str* definiert. In *kc_signon_str* liefert UTM bei KC_GET_OBJECT die Werte der Parameter zurück, über die die Anmeldung der Kommunikationspartner an die Anwendung gesteuert wird.

Datenstruktur kc_signon_str
`char concurrent_terminal_signon[3];`
`char grace;`
`char pw_history[2];`
`char restricted;`
`char silent_alarm[3];`
`char upic;`
`char multi_signon;`
`char omit_upic_signoff;`

Die Felder der Datenstruktur haben folgende Bedeutung:

concurrent_terminal_signon

ist nur relevant, wenn in Ihrer Anwendung ein Anmelde-Vorgang generiert ist.

concurrent_terminal_signon gibt an, für wieviel Prozent der generierten Benutzer gleichzeitig ein Anmelde-Vorgang aktiv sein kann, der für eine Anmeldung über ein Terminal oder eine TS-Anwendung (APPLI oder SOCKET) gestartet wurde.

UTM versucht entsprechend dieser Angabe die notwendigen Betriebsmittel anzulegen.

grace

(Grace-Sign-On)
legt fest, ob ein Benutzer bei der ersten Anmeldung nach Ablauf der Passwort-Gültigkeit (siehe *kc_user_str. protect_pw_time*) sein Passwort noch ändern darf.

'N' Der Benutzer kann sein Passwort nach Ablauf der Gültigkeit nicht mehr ändern. Das Passwort kann nach Ablauf der Gültigkeit nur noch vom Administrator geändert werden.

'Y' Der Benutzer kann sein Passwort nach Ablauf der Gültigkeit noch ändern.
Die Änderung muss innerhalb des Anmeldeverfahrens erfolgen, bevor der Benutzer vollständig angemeldet ist.

Falls ein Anmelde-Vorgang aktiviert ist, dann kann das Passwort dort mit dem KDCS-Aufruf SIGN CP geändert werden, unabhängig vom Client-Typ.
Ein Anmelde-Vorgang wird immer aktiviert, wenn sich ein Benutzer über eine Verbindung anmeldet, für deren Transportzugangspunkt ein Anmelde-Vorgang generiert ist.

Die folgende Tabelle zeigt, wie sich die einzelnen Client-Typen bei abgelaufenem Passwort verhalten und wie das Verhalten davon abhängt, ob ein Anmelde-Vorgang aktiviert ist.

Client-Typ	**Verhalten, wenn das Passwort abgelaufen ist** [1]
UPIC	Das Passwort kann unabhängig von der Aktivierung eines Anmelde-Vorgangs auch mit der Funktion *Set_Conversation_-Security_New_Password* geändert werden.
BS2000-Terminal	Bei dunkelgesteuertem Passwort fordert openUTM den Benutzer zur Änderung des Passwortes auf, unabhängig davon, ob ein Anmelde-Vorgang aktiviert ist.
	Bei nicht-dunkelgesteuertem Passwort fordert openUTM den Benutzer nur dann zur Änderung des Passwortes auf, wenn kein Anmelde-Vorgang aktiviert ist.
Terminal auf Unix-, Linux- und Windows-Systemen	openUTM fordert den Benutzer zur Änderung des Passwortes auf, unabhängig davon, ob ein Anmelde-Vorgang aktiviert ist.
TS-Anwendung	Der Benutzer kann das Passwort ohne Aktivierung eines Anmelde-Vorgangs nicht mehr ändern.

1 Das Passwort kann immer über die Administrations-Schnittstelle geändert werden (z.B. KC_MO-DIFY_OBJECT, *obj_type*=KC_USER). Standardmäßig werden Passwörter mit beschränkter Gültigkeitsdauer bei Änderung über die Administrations-Schnittstelle sofort auf „abgelaufen“ gesetzt. Wollen Sie dies verhindern, müssen Sie dies an der Administrations-Schnittstelle explizit anfordern.

pw_history

gibt an, über wieviele Passwort-Änderungen für jede Benutzerkennung UTM eine Passwort-Historie führt. *pw_history* enthält die Anzahl der Passwörter einer Benutzerkennung, die sich UTM merkt.

Ändert ein Benutzer sein Passwort und ist für das Passwort in der USER-Anweisung eine beschränkte Gültigkeitsdauer generiert, dann muss das neue Passwort sich von dem aktuellen und den letzten *n* Passworten, die für die Benutzerkennung gesetzt wurden, unterscheiden. Dabei ist *n* die Anzahl in *pw_history*.

pw_history=0 bedeutet, dass UTM keine Passwort-Historie führt.

Die Passwort-Historie ist nur bei einer Passwort-Änderung durch den Benutzer von Bedeutung, eine Änderung des Passworts per Administration ist unabhängig von der in der Historie enthaltenen Passwörtern möglich.

restricted

gibt an, ob im 1.Teil des Anmelde-Vorgangs Datenbank-Aufrufe und Zugriffe auf globale UTM-Speicherbereiche verboten sind.

'Y' Im 1. Teil des Anmelde-Vorgangs sind Datenbank-Aufrufe und Zugriffe auf globale UTM-Speicherbereiche verboten.

'N'

Im 1. Teil des Anmelde-Vorgangs sind Datenbank-Aufrufe und Zugriffe auf globale UTM-Speicherbereiche erlaubt.

silent_alarm

gibt an, nach wievielen unmittelbar aufeinanderfolgenden erfolglosen Anmeldeversuchen eines Terminal-Benutzers UTM einen „stillen Alarm“ auslöst. Stiller Alarm bedeutet, dass UTM die Meldung K094 erzeugt. Dieser Wert ist über das Feld *signon_fail* der Datenstruktur *kc_max_par_str* änderbar, siehe Abschnitt "kc_max_par_str - Maximalwerte der Anwendung (MAX-Parameter)" .

upic ist nur relevant, wenn in Ihrer Anwendung ein Anmelde-Vorgang generiert ist.
upic gibt an, ob der Anmelde-Vorgang aktiviert wird, wenn ein UPIC-Client eine Conversation starten will.

'Y' Ist für den Transportsystemzugangspunkt, über den der UPIC-Client die Verbindung aufgebaut hat, ein Anmelde-Vorgang generiert, dann wird dieser vor jeder Conversation gestartet, die von dem UPIC-Client initiiert wird.

'N' Für UPIC-Clients wird kein Anmelde-Vorgang gestartet.

multi_signon

gibt an, ob sich gleichzeitig mehrere Benutzer unter derselben Benutzerkennung bei der Anwendung anmelden dürfen.

'Y' Folgende Fälle sind zu unterscheiden:

- Die Benutzerkennung ist mit RESTART=NO generiert:

  In diesem Fall dürfen sich mehrere Benutzer gleichzeitig mit derselben Benutzerkennung bei der Anwendung anmelden. Dabei darf sich jedoch nur einer der Benutzer am Terminal anmelden.
- Die Benutzerkennung ist mit RESTART=YES generiert:

  In diesem Fall können nur mehrere Auftragnehmer-Vorgänge gleichzeitig unter einer Benutzerkennung aktiv sein, wenn die Auftragnehmer-Vorgänge über OSI TP-Verbindungen gestartet wurden und die Funktion „Commit“ ausgewählt wurde.

'N' Mit jeder Benutzerkennung der Anwendung darf höchstens ein Benutzer einmal angemeldet sein. D.h. für jede Benutzerkennung kann zu einer Zeit maximal ein Dialog-Vorgang aktiv sein und wenn ein Benutzer an die Anwendung angemeldet ist, dann kann für diese Benutzerkennung auch kein Auftragnehmer-Vorgang in dieser Anwendung gestartet werden.

> **i** *multi_signon* hat keine Auswirkung auf das Anmelden eines Benutzers über eine OSI TP-Verbindung zur Erzeugung eines Asynchron-Auftrags.

omit_upic_signoff

gibt an, ob die Benutzerkennung, unter der sich ein UPIC-Client anmeldet, nach Ende einer UPIC-Conversation angemeldet bleibt oder nicht.

'Y' Die Benutzerkennung bleibt nach Ende einer UPIC-Conversation angemeldet. Sie wird erst wieder abgemeldet,

- wenn die Verbindung abgebaut wird oder
- wenn sich über dieselbe Verbindung ein UPIC-Client mit einer anderen Benutzerkennung anmelden will.

Wird vom UPIC-Client keine andere Benutzerkennung übergeben, bleibt die ursprüngliche Benutzerkennung angemeldet, d.h. vor dem Start der neuen UPIC-Conversation wird kein Anmeldevorgang gestartet.

Bei Anwendungen ohne Benutzerkennungen wird gegebenenfalls vor dem Start der ersten UPIC-Conversation nach Verbindungsaufbau einmal ein Anmeldevorgang gestartet.

Standard in UTM-Cluster-Anwendungen.

'N' Die Benutzerkennung, mit der sich ein UPIC-Client angemeldet hat, wird am Ende jeder UPIC-Conversation abgemeldet.

Standard in stand-alone UTM-Anwendungen.

11.3.2.11 kc_system_par_str - Systemparameter

Für den Parametertyp KC_SYSTEM_PAR ist die Datenstruktur *kc_system_par_str* definiert. In *kc_system_par_str* liefert UTM bei KC_GET_OBJECT folgende Informationen zurück:

- Basiseinstellungen der Anwendung, z.B. die Anwendung ist generiert für die Server-Server-Kommunikation.
- openUTM-Version mit Korrekturstand.
- Anwendungsname und Funktionsvariante.
- Betriebssystem sowie Name, Plattform und Modus des Rechners, auf dem die Anwendung abläuft.

Datenstruktur kc_system_par_str
`char appliname[8];`
`char utm_version[8];`
`char applimode;`
`char system_type;`
`char hostname[8];`
`char destadm[8];`
`char tacclasses;`
`char pgwt;`
`char kdcload;` (nur auf BS2000-Systemen)
`char load_module_gen;`
`char prog_change_running;`
`char inverse_kdcdef_state;`
`char utmd;`
`char osi_tp;`
`char certificate_gen;` (nur auf BS2000-Systemen)
`char os[24];`
`char bit_mode[8];`
`char cluster_appl;`
`char hostname_long[64];`

Die Felder der Datenstruktur haben die folgende Bedeutung:

appliname

Name der Anwendung, der bei der KDCDEF-Generierung in MAX APPLINAME festgelegt wurde.

utm_version

eingesetzte openUTM-Version einschließlich Korrekturstand, z.B. V07.0A00.

applimode

gibt an, ob es sich bei der UTM-Anwendung um eine UTM-S- oder UTM-F-Anwendung handelt.

'S' Die Anwendung ist als UTM-S-Anwendung generiert (Secure).

'F' Die Anwendung ist als UTM-F-Anwendung generiert (Fast).

system_type

Betriebssystem des Rechners, auf dem die Anwendung abläuft.

'B' BS2000-Systeme

'X' Unix- und Linux-Systeme

'N' Windows-Systeme

hostname

Name des Rechners, auf dem die Anwendung abläuft.

Hinweis für Unix-, Linunx- und Windows-Systeme:

Ist dieser Name länger als 8 Zeichen, dann kann der vollständige, bis zu 64 Zeichen lange Name dem Feld *hostname_long* entnommen werden. Das Feld *hostname* enthält in diesem Fall die ersten 8 Zeichen des langen Namens.

destadm

enthält den Empfänger, an den UTM die Ergebnisse von KDCADM-Administrationsaufrufen sendet, die asynchron verarbeitet wurden (Asynchron-Transaktionscodes von KDCADM). *destadm* kann Folgendes enthalten:

- den Namen eines LTERM-Partners oder
- den Transaktionscode eines Asynchron-Teilprogramms.

Enthält *destadm* Leerzeichen, dann ist kein Empfänger definiert. Die Ergebnisse der Asynchron-Transaktionscodes von KDCADM gehen verloren.

tacclasses

gibt an, ob die Anwendung mit TAC-Klassen generiert wurde, d.h. ob bei der KDCDEF-Generierung TAC-Klassen erzeugt wurden.

'Y' Die Anwendung wurde mit TAC-Klassen generiert.

'N' Die Anwendung wurde ohne TAC-Klassen generiert.

pgwt

gibt an, ob in der Anwendung Teilprogrammläufe mit blockierenden Aufrufen (z.B. KDCS-Aufruf PGWT) erlaubt sind.

'Y' In der Anwendung sind blockierende Aufrufe erlaubt, d.h. es gibt entweder mindestens einen Transaktionscode oder eine TAC-Klasse mit der Eigenschaft *pgwt*='Y' (siehe *kc_tac_str.pgwt* im Abschnitt "kc_tac_str - Transaktionscodes lokaler Services" und *kc_tacclass_str.pgwt* im Abschnitt "kc_tacclass_str - TAC-Klassen der Anwendung").

'N' Es sind keine blockierenden Aufrufe erlaubt, d.h. die Anwendung enthält weder Transaktionscodes noch TAC-Klassen, für die *pgwt*='Y' gesetzt ist.

kdcload (nur auf BS2000-Systemen)

Dieses Feld enthält immer 'N'.
Dieses Feld bezieht sich auf eine nicht mehr unterstützte Funktionalität von UTM.

load_module_gen

gibt an, ob die Anwendung mit Lademodulen (BS2000-Systeme) bzw. mit Shared Objects/DLLs (Unix-, Linux- und Windows-Systeme) generiert wurde. D.h. bei der KDCDEF-Generierung wurde mindestens eine LOAD-MODULE-Anweisung bzw. eine SHARED-OBJECT-Anweisung angegeben.

'Y' Die Anwendung wurde mit LOAD-MODULE- bzw. SHARED OBJECT-Anweisungen generiert.

'N' Die Anwendung wurde nicht mit LOAD-MODULE- bzw. SHARED OBJECT-Anweisungen generiert.

prog_change_running

gibt an, ob UTM gerade einen Programmaustausch für die Anwendung durchführt.

'Y' Es läuft gerade ein Programmaustausch.

'N' Es läuft kein Programmaustausch.

inverse_kdcdef_state

gibt an, ob z.Zt. ein inverser KDCDEF läuft, d.h. ein KC_CREATE_STATEMENTSAufruf bearbeitet wird.

'N' Es läuft z.Zt. kein inverser KDCDEF.

'A' Der Lauf eines inversen KDCDEF ist vorbereitet. Er wird asynchron gestartet, sobald alle Transaktionen, die Konfigurationsdaten ändern, beendet sind. Administrationsaufrufe, die die Konfigurationsdaten verändern, werden abgewiesen.

'Y' Es läuft gerade ein inverser KDCDEF.

utmd

gibt an, ob die Anwendung für die verteilte Verarbeitung über ein höheres Kommunikationsprotokoll (LU6.1 oder OSI TP) generiert ist.

'Y' Die Anwendung wurde für die verteilte Verarbeitung generiert.

'N' Die Anwendung wurde nicht für die verteilte Verarbeitung generiert.

osi_tp

gibt an, ob die Anwendung für die verteilte Verarbeitung über OSI TP generiert ist.

'Y' Die Anwendung wurde mit Anweisungen für OSI TP generiert.

'N' Die Anwendung wurde ohne Anweisungen für OSI TP generiert.

certificate_gen (nur auf BS2000-Systemen)

Dieser Parameter wird nicht mehr unterstützt.

os Gibt die Systemplattform des Rechners an, z.B. 'Windows Intel' oder 'Solaris Sparc'.

bit_mode

Modus, in dem das Betriebssystem abläuft:

'32 Bit' 32-bit-Modus

'64 Bit' 64-bit-Modus

cluster_appl

gibt an, ob die Anwendung zu einer UTM-Cluster-Anwendung gehört.

'Y' Anwendung ist Knoten-Anwendung einer UTM-Cluster-Anwendung.

'N' Anwendung ist eine stand-alone UTM-Anwendung.

hostname_long

Name des Rechners, auf dem die Anwendung abläuft.

11.3.2.12 kc_tasks_par_str - Anzahl der Prozesse

Für den Parametertyp KC_TASKS_PAR ist die Datenstruktur *kc_tasks_par_str* definiert. In *kc_tasks_par_str* liefert UTM bei KC_GET_OBJECT alle Informationen zu den Prozessen der Anwendung zurück:

- Maximalzahl und momentan eingestellte Anzahl der Prozesse der Anwendung.
- Maximalzahl der Prozesse, die gleichzeitig Asynchron-Aufträge bearbeiten dürfen.
- in wievielen der Prozesse gleichzeitig Teilprogramme mit blockierenden Aufrufen (z.B. PGWT) ablaufen dürfen.
- Anzahl der Prozesse, die für die Bearbeitung von UTM-internen Aufgaben und Dialog-Aufträgen, die keiner Dialog-TAC-Klasse zugeordnet sind, freigehalten werden. Diese Prozesszahl wird nur zurückgeliefert, wenn in der Anwendung die Bearbeitung von Aufträgen prioritätengesteuert erfolgt d.h. bei der KDCDEF-Generierung die Anweisung TAC-PRIORITIES abgesetzt wurde.

mod[1]	**Datenstruktur kc_tasks_par_str**
-	`char tasks[3];`
-	`char asyntasks[3];`
-	`char tasks_in_pgwt[3];`
x(A)	`char mod_max_tasks[3];`
x(A)	`char mod_max_asyntasks[3];`
x(A)	`char mod_max_tasks_in_pgwt[3];`
-	`char curr_max_asyntasks[3];`
-	`char curr_max_tasks_in_pgwt[3];`
-	`char curr_tasks[3];`
-	`char curr_asyntasks[3];`
-	`char curr_tasks_in_pgwt[3];`
x(A)	`char mod_free_dial_tasks[3];`
-	`char gen_system_tasks[3];`
-	`char curr_system_tasks[3];`

1 Feldinhalt mit KC_MODIFY_OBJECT modifizierbar; siehe Abschnitt "obj_type=KC_TASKS_PAR"

Die Felder der Datenstruktur haben die folgende Bedeutung:

tasks

generierter Grenzwert für die maximale Anzahl der Prozesse, die gleichzeitig für die Anwendung eingesetzt werden können.

Die maximale Anzahl der Prozesse, die Aufträge der Anwendung bearbeiten, wird bei jedem Start der Anwendung festgelegt und kann während des Anwendungslaufs an die aktuellen Erfordernisse angepasst werden (siehe *mod_max_tasks*).

Weder die beim Start angegebene Anzahl der Prozesse noch die durch die Administration eingestellte Maximalzahl kann den Wert von *tasks* überschreiten.

asyntasks

generierter Grenzwert für die maximale Anzahl der Prozesse der Anwendung, die gleichzeitig für die Asynchron-Verarbeitung verwendet werden können.
Die für den aktuellen Anwendungslauf gewünschte Maximalzahl der Prozesse zur Bearbeitung von Asynchron-Aufträgen kann beim Start der Anwendung oder durch die Administration eingestellt werden (siehe Feld *mod_max_asyntasks*), diese Zahl darf den Wert von *asyntasks* nicht überschreiten.

tasks_in_pgwt

generierter Grenzwert für die maximale Anzahl der Prozesse, in denen gleichzeitig Teilprogramme mit blockierenden Aufrufen ablaufen dürfen (z.B. KDCS-Aufruf PGWT; Program Wait).
Die für den aktuellen Anwendungslauf gewünschte Maximalzahl kann beim Start der Anwendung oder durch die Administration eingestellt werden (siehe Feld *mod_max_tasks_in_pgwt*). Diese Zahl darf den Wert von *tasks_in_pgwt* nicht überschreiten.

mod_max_tasks

enthält die aktuell eingestellte maximale Anzahl der Prozesse, die insgesamt gleichzeitig für die Anwendung eingesetzt werden können. *mod_max_tasks* enthält die Anzahl, die zuletzt festgesetzt wurde, entweder beim Start der Anwendung oder durch die Administration (z.B. KC_MODIFY_OBJECT mit KC_TASKS_PAR).

mod_max_tasks enthält den Sollwert für die aktuelle Prozessanzahl. Die Anzahl der momentan tatsächlich aktiven Prozesse, die aktuell Aufträge der Anwendung bearbeiten können, ist im Feld *curr_tasks* abgelegt. Diese kann nur beim Starten oder Beenden eines Prozesses kurzfristig anders sein als *mod_max_tasks*.

Maximalwert: *tasks*
Minimalwert: '1'

mod_max_asyntasks

aktuell eingestellter Grenzwert für die Anzahl der Prozesse, die gleichzeitig für Asynchron-Verarbeitung verwendet werden dürfen. *mod_max_asyntasks* enthält die Anzahl der Prozesse für Asynchron-Verarbeitung, die entweder beim Start der Anwendung oder durch die Administration festgelegt wurde (z. B. KC_MODIFY_-OBJECT mit KC_TASKS_PAR).

Die tatsächliche Maximalzahl der Prozesse, die gleichzeitig für Asynchron-Verarbeitung verwendet werden dürfen (*curr_max_asyntasks*), kann kleiner sein als der in *mod_max_asyntasks* angegebene Wert, da die tatsächliche Anzahl durch die Anzahl der aktuell laufenden Prozesse der Anwendung (*curr_tasks*) begrenzt ist.

mod_max_asyntasks entspricht einem momentanen oberen Grenzwert.

Minimalwert: '0'
Maximalwert: Anzahl in *asyntasks*

mod_max_tasks_in_pgwt

aktuell eingestellter Grenzwert für die Anzahl der Prozesse, in denen gleichzeitig Teilprogramme mit blockierenden Aufrufen (Program Wait; z.B. der KDCS-Aufruf PGWT) ablaufen dürfen. *mod_max_tasks_in_pgwt* enthält die Anzahl, die zuletzt beim Start der Anwendung oder durch die Administration gesetzt wurde (z.B. KC_MODIFY_OBJECT mit KC_TASKS_PAR).

Die tatsächliche Maximalzahl der Prozesse, die gleichzeitig Teilprogramme mit blockierenden Aufrufen bearbeiten (*curr_max_tasks_in_pgwt*), kann kleiner sein als der in *mod_max_tasks_in_pgwt* angegebene Wert, da die tatsächliche Anzahl mindestens 1 kleiner sein muss als die Anzahl der aktuell laufenden Prozesse der Anwendung (*curr_tasks*).

mod_max_tasks_in_pgwt entspricht einem momentanen oberen Grenzwert.

Minimalwert: '0'
Maximalwert: Anzahl in *tasks_in_pgwt*

curr_max_asyntasks

aktuelle Maximalzahl der Prozesse, die gleichzeitig für Asynchron-Verarbeitung verwendet werden dürfen. Diese Prozesszahl ist der kleinere Wert von dem aktuell eingestellten Grenzwert der Prozesse, die gleichzeitig für Asynchron-Verarbeitung verwendet werden dürfen (*mod_max_asyntasks*), und der Anzahl aktuell laufender Prozesse der Anwendung (*curr_tasks*). *curr_max_asyntasks* wird von UTM dynamisch angepasst, wenn einer der beiden Werte *curr_tasks* oder *mod_max_asyntasks* geändert wird. Siehe Abschnitt "Mögliche Maßnahmen".

curr_max_tasks_in_pgwt

aktuelle maximale Anzahl der Prozesse, in denen gleichzeitig Teilprogramme mit blockierenden Aufrufen ablaufen dürfen (KDCS-Aufruf PGWT). Diese Prozesszahl ist der kleinere Wert von dem aktuell eingestellten Grenzwert der Prozesse, in denen gleichzeitig Teilprogramme mit blockierenden Aufrufen ablaufen dürfen (*mod_max_tasksk_in_pgwt*), und der Anzahl aktuell laufender Prozesse der Anwendung (*curr_tasks*) minus eins. *curr_max_asyntasks* wird von UTM dynamisch angepasst, wenn einer der beiden Werte *curr_tasks* oder *mod_max_tasks_in_pgwt* geändert wird. Siehe Abschnitt "Mögliche Maßnahmen".

curr_tasks

enthält die Anzahl der momentan laufenden Prozesse der Anwendung. Der Wert von *curr_tasks* entspricht im Normalfall dem Wert von *mod_max_tasks*. Der Wert von *curr_tasks* kann jedoch temporär kleiner oder größer als *mod_max_tasks* sein. Er ist kleiner, wenn ein Prozess sich abnormal beendet hat und noch nicht wieder automatisch gestartet worden ist. Er kann größer sein, wenn kurz vorher die Soll-Prozesszahl in *mod_max_tasks* herabgesetzt wurde. *curr_tasks* enthält den Istwert der Prozessanzahl, *mod_max_tasks* den Sollwert.

curr_asyntasks

enthält die Anzahl der Prozesse, die momentan Asynchron-Aufträge bearbeiten.

curr_tasks_in_pgwt

enthält die Anzahl der Prozesse, in denen momentan Teilprogramme mit blockierenden Aufrufen (z.B. PGWT) ablaufen, d.h. Anzahl der momentan im Program Wait wartenden Prozesse.

mod_free_dial_tasks

nur relevant, wenn bei der KDCDEF-Generierung eine TAC-PRIORITIES-Anweisung abgesetzt wurde.

In *mod_free_dial_tasks* liefert UTM die aktuell eingestellte Anzahl an Prozessen zurück, die für die Bearbeitung von UTM-internen Aufgaben und für Aufträge, die keiner Dialog-TAC-Klasse zugeordnet sind, freigehalten werden. Dieser Anteil der Gesamtprozesszahl steht dann nicht für die Bearbeitung von Aufträgen an Dialog-TAC-Klassen zur Verfügung.

Wird die maximale Anzahl der Prozesse der Anwendung herabgesetzt, so dass sie kleiner oder gleich *mod_free_dial_tasks* ist, dann bearbeitet trotzdem ein Prozess Aufträge an Dialog-TAC-Klassen.

Minimalwert: '0'
Maximalwert: Wert in *tasks* -1

Ist die Anwendung ohne TAC-PRIORITIES generiert, dann liefert UTM in *mod_free_dial_tasks* Leerzeichen zurück.

gen_system_tasks

enthält die maximale Anzahl der UTM-System-Prozesse, die aufgrund der aktuellen Konfiguration gestartet werden können.

curr_system_tasks

enthält die Anzahl der momentan laufenden UTM-System-Prozesse.

11.3.2.13 kc_timer_par_str - Timer-Einstellungen

Für den Parametertyp KC_TIMER_PAR ist die Datenstruktur *kc_timer_par_str* definiert. In *kc_timer_par_str* liefert UTM bei KC_GET_OBJECT die aktuelle Einstellung aller Timer der UTM-Anwendung zurück.

Die Timer werden bei der Generierung der Anwendung festgelegt und können während des Anwendungslaufs an die aktuelle Situation angepasst werden, mit Operationscode KC_MODIFY_OBJECT oder mit Hilfe des Administrationskommandos KDCAPPL.

Die Änderung eines Timers wirkt erst dann, wenn der Timer das erste Mal nach der Änderung neu aufgezogen wird. Die Änderung ist nicht wirksam für bereits laufende Timer. Die Änderungen wirken nur während des aktuellen Anwendungslaufs.

mod[1]	**Datenstruktur kc_timer_par_str**
x(GIR)	`char conrtime_min[5];`
x(GIR)	`char pgwttime_sec[5];`
x(GIR)	`char reswait_ta_sec[5];`
x(GIR)	`char reswait_pr_sec[5];`
x(GIR)	`char termwait_in_ta_sec[5];`
-	`char termwait_end_ta_sec[5];`
x(GIR)	`char logackwait_sec[5];` (nur auf BS2000-Systemen)
x(GIR)	`char conctime1_sec[5];`
x(GIR)	`char conctime2_sec[5];`
x(GIR)	`char ptctime_sec[5];`
-	`char qtime1[5];`
-	`char qtime2[5];`

1 Feldinhalt mit KC_MODIFY_OBJECT modifizierbar; siehe Abschnitt "obj_type = KC_TIMER_PAR"

Die Felder der Datenstruktur haben die folgende Bedeutung:

conrtime_min

(**con**nection **r**equest **time**)
Zeit in Minuten, nach der UTM beim Ausfall einer Verbindung zu einer Partner-Anwendung bzw. zu einem Client oder Drucker versuchen soll, diese Verbindung wieder aufzubauen. *conrtime* bezieht sich auf Verbindungen zu:

- Druckern, zu denen UTM beim Start der Anwendung automatisch eine Verbindung aufbaut (*auto_connect*='Y' in *kc_pterm_str*, "kc_pterm_str - Clients und Drucker"), wenn diese Verbindung nicht zuvor durch die Administration abgebaut wurde.
- Druckern, zu denen UTM eine Verbindung aufbaut, sobald die Anzahl der Druckaufträge für diesen Drucker den generierten Schwellwert überschreitet (LTERM-Partner mit *plev* > 0). UTM versucht nur

dann die Verbindung wieder aufzubauen, wenn die Anzahl der nach dem Verbindungsabbruch noch anstehenden Druckaufträge größer oder gleich dem Schwellwert ist.
Ist *conrtime_min* != 0, dann versucht UTM auch dann die Verbindung zu dem Drucker aufzubauen, wenn die Verbindung zuvor explizit durch die Administration abgebaut wurde.

- TS-Anwendungen (*ptype*='APPLI' oder 'SOCKET' in *kc_pterm_str*), zu denen UTM beim Start der Anwendung automatisch eine Verbindung aufbaut (*auto_-connect*='Y' in *kc_pterm_str*, "kc_pterm_str - Clients und Drucker"), wenn diese Verbindung nicht zuvor durch die Administration abgebaut wurde.
- OSI TP- oder LU6.1-Partner-Anwendungen, zu denen UTM beim Start der Anwendung automatisch eine Verbindung aufbaut, wenn diese Verbindung nicht zuvor durch die Administration oder von UTM wegen Ablauf eines Timers (*idletime*) abgebaut wurde.
- Nachrichtenverteilern (MUX) auf BS2000-Sytemen, zu denen UTM beim Start der Anwendung automatisch eine Verbindung aufbaut, wenn diese Verbindung nicht zuvor durch die Administration abgebaut wurde.

Kommt bei Partnern, die mit automatischem Verbindungsaufbau konfiguriert sind, keine Verbindung zustande (beim Anwendungsstart oder nach einer Verbindungsanforderung durch die Administration) oder wird die Verbindung unterbrochen, so versucht UTM abhängig vom Grund des Verbindungsverlustes im Abstand von *conrtime_min* die Verbindung neu- bzw. wiederaufzubauen.

Bei *conrtime_min*='0' unternimmt UTM keinen Versuch, eine ausgefallene Verbindung wieder aufzubauen.

Maximalwert: '32767'
Minimalwert: '0'

pgwttime_sec

Zeit in Sekunden, die ein Teilprogramm nach einem blockierenden Funktionsaufruf maximal auf das Eintreffen von Nachrichten warten darf. Blockierende Funktionsaufrufe sind Aufrufe, bei denen die Kontrolle erst nach Eintreffen der Antwort an das Teilprogramm zurückgegeben wird (z.B. KDCS-Aufruf PGWT).

Während dieser Wartezeit bleibt ein Prozess der UTM-Anwendung exklusiv für dieses Teilprogramm reserviert.

Maximalwert: '32767'
Minimalwert: '60'

reswait_ta_sec

Zeit in Sekunden, die ein Teilprogramm maximal auf ein von einer anderen Transaktion gesperrtes Betriebsmittel wartet: z.B. GSSBs, ULSs, TLSs.

Ist nach Ablauf dieser Zeit das Betriebsmittel nicht verfügbar, erhält das Anwendungsprogramm einen entsprechenden Returncode KCRCCC.

Die in *reswait_ta_sec* angegebene Wartezeit hat keine Bedeutung, wenn die Sperre von einer Mehrschritt-Transaktion gehalten wird, die auf eine Eingabe-Nachricht wartet (nach PEND KP oder PGWT KP). In diesem Fall erhalten alle Teilprogramme,
die auf die gesperrten Betriebsmittel zugreifen, sofort (ohne Wartezeit *reswait_ta_sec*) einen Returncode KCRCCC.

reswait_ta_sec='0' bedeutet, dass nicht gewartet wird. Ein Teilprogrammlauf, der auf gesperrte Betriebsmittel zugreifen will, erhält sofort einen entsprechenden Returncode.

Maximalwert: '32767'
Minimalwert: '0'

reswait_pr_sec

Zeit in Sekunden, die maximal auf ein von einem anderen Prozess gesperrtes Betriebsmittel gewartet wird. Wird diese Zeit überschritten, dann beendet sich die Anwendung mit einer Fehlermeldung (siehe KC_MODIFY_OBJECT, *obj_type*=KC_TIMER_PAR im Abschnitt "obj_type = KC_TIMER_PAR").

Maximalwert: '32767'
Minimalwert: '300'

termwait_in_ta_sec

Zeit in Sekunden, die bei einer Mehrschritt-Transaktion (d.h. im Programm PEND KP) maximal zwischen einer Ausgabe an einen Dialog-Partner und der nachfolgenden Dialog-Antwort verstreichen darf. Wird die Zeit *termwait_in_ta_sec* überschritten, dann wird die Transaktion zurückgesetzt. Die von der Transaktion reservierten Betriebsmittel werden freigegeben. Die Verbindung zum Partner wird abgebaut.

Maximalwert: '32767'
Minimalwert: '60'

termwait_end_ta_sec

enthält keinen gültigen Wert. Die Zeit in Sekunden, die UTM nach dem Ende einer Transaktion oder nach dem Anmelden (KDCSIGN) maximal auf eine Eingabe vom Dialog-Partner wartet, wird ab UTM V5.0 Partner-spezifisch festgelegt. Informationen über diesen Timer erhalten Sie, wenn Sie KC_GET_OBJECT mit *obj_type* KC_PTERM bzw. KC_TPOOL aufrufen (Feld *idletime* im Abschnitt "kc_pterm_str - Clients und Drucker" und "kc_tpool_str - LTERM-Pools der Anwendung").

logackwait_sec (nur auf BS2000-Systemen)

Zeit in Sekunden, die UTM maximal auf eine logische Abdruckquittung vom Drucker bzw. auf eine Transportquittung für eine Asynchron-Nachricht an eine andere Anwendung (erzeugt mit dem KDCS-Aufruf FPUT) warten soll.

Trifft nach dieser Zeit die Quittung nicht ein, z.B. wegen Papierende bei Druckern, baut UTM die logische Verbindung zu dem Gerät ab.

Minimalwert: '10'
Maximalwert: '32767'

Folgende Timer haben nur bei UTM-Anwendungen mit verteilter Verarbeitung über LU6.1 bzw. OSI TP eine Bedeutung.

conctime1_sec

(**con**nection **c**ontrol **time**)
Zeit in Sekunden zur Überwachung des Aufbaus einer Session (LU6.1) bzw. Association (OSI TP). Wird die Session bzw. Association innerhalb der angegebenen Zeit nicht aufgebaut, dann baut UTM die Transportverbindung zur Partner-Anwendung ab. Damit wird verhindert, dass eine Transportverbindung wegen eines misslungenen Aufbaus einer Session bzw. Association blockiert bleibt. Das ist möglich, wenn eine zum Aufbau erforderliche Nachricht verloren geht.

conctime1_sec='0' bedeutet bei LU6.1-Verbindungen, dass der Aufbau einer Session nicht überwacht wird (es wird beliebig lange gewartet), und bei OSI TP-Verbindungen, dass maximal 60 Sekunden auf den Aufbau einer Association gewartet wird.

Minimalwert: '0'
Maximalwert: '32767'

conctime2_sec

Zeit in Sekunden, die beim Übertragen einer Asynchron-Nachricht maximal auf eine Quittung vom Empfänger gewartet wird. Nach Ablauf der Zeit *conctime2_sec* baut UTM die Transportverbindung ab. Der Asynchron-Auftrag geht nicht verloren, er steht weiterhin in der lokalen Message Queue. Mit der Überwachung wird vermieden, dass eine Verbindung nicht genutzt werden kann, weil eine Quittung verloren ging, oder ein Verbindungsverlust nicht vom Transportsystem an UTM gemeldet wurde.

conctime2_sec='0' bedeutet, dass keine Überwachung durchgeführt wird.

Minimalwert: '0'
Maximalwert: '32767'

ptctime_sec

hat nur eine Bedeutung für die verteilte Verarbeitung über LU6.1-Verbindungen. *ptctime_sec* legt die Zeit in Sekunden fest, die ein Auftragnehmer-Vorgang maximal im Zustand PTC (prepare to commit, Transaktionsstatus P) auf eine Quittung vom Auftraggeber-Vorgang wartet. Nach Ablauf der Zeit wird die Verbindung zum Auftraggeber abgebaut, die Transaktion im Auftragnehmer-Vorgang zurückgesetzt und der Vorgang beendet. Dadurch kann es evtl. zu einem Mismatch kommen.

Wurde für die Anwendung bereits KDCSHUT WARN oder GRACE gegeben und der Wert von *ptc_time_sec* ist ungleich 0, so wird die Wartezeit unabhängig von *ptc_time_sec* so gewählt, dass die Transaktion zurückgesetzt wird, bevor die Anwendung beendet wird, um eine abnormale Anwendungsbeendigung mit ENDPET möglichst zu vermeiden.

ptctime_sec='0' bedeutet, dass beliebig lange auf eine Quittung gewartet wird.

Minimalwert: '0'
Maximalwert: '32767'

qtime1

zeigt die Zeit in Sekunden an, die ein Dialog-Vorgang maximal auf das Eintreffen von Nachrichten für USER-, TAC- oder temporäre Queues warten darf.

qtime2

zeigt die Zeit in Sekunden an, die ein Asynchron-Vorgang maximal auf das Eintreffen von Nachrichten für USER-, TAC- oder temporäre Queues warten darf.

11.3.2.14 kc_utmd_par_str - Parameter für die verteilte Verarbeitung

Für den Parametertyp KC_UTMD_PAR ist die Datenstruktur *kc_utmd_par_str* definiert. In *kc_utmd_par_str* liefert UTM bei KC_GET_OBJECT die Basiseinstellungen für die verteilte Verarbeitung über LU6.1 und OSI TP zurück.

Datenstruktur kc_utmd_par_str
`char application_process_title[10][8];`
`char maxjr[3];`
`char rset;`

Die Felder der Datenstruktur haben die folgende Bedeutung:

application_process_title

ist nur für die verteilte Verarbeitung über OSI TP relevant.
application_process_title enthält den Application Process Title der lokalen Anwendung (siehe openUTM-Handbuch „Anwendungen generieren“).

Ein Application Process Title besteht aus mindestens 2, maximal aber aus 10 Komponenten. Jede einzelne Komponente ist eine positive ganze Zahl, maximal 8 Zeichen lang.

UTM liefert pro Komponente des Application Process Title ein Feldelement zurück, d.h. die Anzahl der belegten Feldelemente in *application_process_title* entspricht der generierten Anzahl der Komponenten. Die restlichen Feldelemente sind mit binär null versorgt.

Wurde kein Application Process Title generiert, dann sind alle Feldelemente von *application_process_title* mit binär null versorgt.

maxjr (**max**imal number of **j**ob **r**eceivers)
gibt an, wieviele ferne Auftragnehmer-Vorgänge maximal gleichzeitig innerhalb der lokalen Anwendung adressiert sein dürfen.

Der Prozentwert bezieht sich auf die Gesamtzahl aller generierten Sessions und Associations (=100%). Der Prozentwert muss zwischen 0 und 200 liegen.

Ein Wert größer 100 bedeutet, dass openUTM APRO-Aufrufe zum Adressieren ferner Vorgänge akzeptiert, auch wenn zu diesem Zeitpunkt (noch) keine Session bzw. Association für diesen Auftrag frei ist.

rset gibt an, wie sich bei der verteilten Verarbeitung das Rücksetzen einer lokalen Transaktion auf die verteilte Transaktion auswirkt.

Eine lokale Transaktion kann zurückgesetzt werden durch einen RSET-Aufruf aus einem Teilprogramm oder durch das Rücksetzen einer Datenbank-Transaktion, die an der lokalen Transaktion beteiligt ist.
rset kann folgende Werte annehmen:

'G' (GLOBAL)
Nach dem Rücksetzen einer lokalen Transaktion muss das Teilprogramm so beendet werden, dass UTM die verteilte Transaktion zurücksetzt.

'L' (LOCAL)
Das Rücksetzen einer lokalen Transaktion hat keinen Einfluss auf die verteilte Transaktion.

Es kann zu Inkonsistenzen in den verteilten Datenbeständen kommen, wenn einige der an einer verteilten Transaktion beteiligten lokalen Transaktionen zurückgesetzt und andere vorgesetzt werden. Ist *rset*='L', dann ist die globale Datenkonsistenz nicht mehr von den beteiligten Systemkomponenten garantiert. Sie liegt dann in der Verantwortung der Anwendungsteilprogramme. Sie müssen entscheiden, in welchen Situationen die verteilte Transaktion noch sinnvoll beendet werden kann und in welchen Situationen sie zurückgesetzt werden muss.

12 Administrationskommandos - KDCADM

In diesem Kapitel sind die Administrationskommandos von openUTM beschrieben, die die Basis-Administrationsfunktionen zur Verfügung stellen. Die Administrationskommandos sind Transaktionscodes des Administrationsprogramms KDCADM, das zusammen mit openUTM ausgeliefert wird. Damit Sie die Administrationskommandos benutzen können, müssen Sie sowohl KDCADM als auch die Administrationskommandos bei der KDCDEF-Generierung in die Konfiguration aufnehmen. In der folgenden Tabelle finden Sie alle Administrationskommandos.

Jedes Administrationskommando gibt es in zwei Ausprägungen:

- ein Kommando, mit dem Sie die Bearbeitung im Dialog veranlassen können
- ein Kommando für die Administration über Message Queuing (Asynchron-Verarbeitung).

Im Folgenden sind die Kommandos für die Administration im Dialog beschrieben.
Die Beschreibung erfolgt in alphabetisch aufsteigender Reihenfolge der Kommandonamen. Die zugehörigen Kommandos für die Administration über Message Queuing haben dieselben Operanden und dasselbe Eingabeformat. Die Kommandoeingabe unterscheidet sich nur durch den anderen Kommandonamen.

Bei der Administration im Dialog liefert openUTM das Ergebnis der Kommandobearbeitung an den Auftraggeber zurück (Benutzer am Terminal, UPIC-Client, TS-Anwendung oder Partner-Anwendung).

Bei den Asynchron-Kommandos werden alle Ergebnisse als asynchrone Nachrichten an einen festen Empfänger (DESTADM) übergeben. Der Empfänger kann sein:

- ein LTERM-Partner (Ausnahme: UPIC-LTERM-Partner sind nicht zulässig)
- der TAC eines Asynchron-Programms
- eine TAC-Queue (Type=Q)

Der Empfänger wird bei der KDCDEF-Generierung in MAX DESTADM= festgelegt und kann über die Programmschnittstelle zur Administration geändert werden, siehe im Abschnitt "obj_type = KC_MAX_PAR". Ist kein Empfänger definiert, geht das Ergebnis verloren. Ist ein TAC angegeben, der das Ergebnis nicht entgegennehmen kann, z.B. ein gesperrter Asynchron-TAC, dann führt openUTM das Administrationskommando nicht aus und schreibt die Meldung K076 in die System-Protokolldatei SYSLOG und nach SYSOUT (auf BS2000-Systemen) bzw. *stderr* (auf Unix-, Linux- und Windows-Systemen).

Liste der Administrationskommandos

Dialogkommando	Asynchron-Kommando
KDCAPPL - Eigenschaften und Grenzwerte für den Betrieb ändern	KDCAPPLA
KDCBNDL - Master-LTERMs austauschen	KDCBNDLA
KDCDIAG - Diagnosehilfen ein- und ausschalten	KDCDIAGA
KDCHELP - Syntax der Administrationskommandos abfragen	KDCHELPA
KDCINF - Informieren über Objekte und Anwendungsparameter	KDCINFA
KDCLOG - Benutzer-Protokolldatei umschalten	KDCLOGA
KDCLPAP - Verbindungen zu (OSI-)LPAP-Partnern administrieren	KDCLPAPA
KDCLSES - Verbindungen für LU6.1-Sessions auf-/abbauen	KDCLSESA
KDCLTAC - Eigenschaften von LTACs ändern	KDCLTACA
KDCLTERM - Eigenschaften von LTERM-Partnern ändern	KDCLTRMA
KDCMUX - Eigenschaften von Multiplex-Anschlüssen ändern (BS2000-Systeme)	KDCMUXA
KDCPOOL - LTERM-Pools administrieren	KDCPOOLA
KDCPROG - Lademodule/Shared Objects/DLLs austauschen	KDCPROGA
KDCPTERM - Eigenschaften von Clients und Druckern ändern	KDCPTRMA
KDCSEND - Nachricht an LTERM-Partner senden (BS2000-Systeme)	KDCSENDA
KDCSHUT - Anwendungslauf beenden	KDCSHUTA
KDCSLOG - SYSLOG-Datei administrieren	KDCSLOGA
KDCSWTCH - Zuordnung Clients, Drucker zu LTERM-Partnern ändern	KDCSWCHA
KDCTAC - Transaktionscodes und TAC-Queues sperren, wieder freigeben	KDCTACA
KDCTCL - Prozess-Anzahl einer TAC-Klasse ändern	KDCTCLA
KDCUSER - Benutzereigenschaften ändern	KDCUSERA

Kommandos von KDCADM

Kommando-Format

```
kommando-name operand1,operand2,...
```

Es gibt Stellungsoperanden und Schlüsselwortoperanden.

Stellungsoperanden sind Angaben ohne Schlüsselwort und „=“-Zeichen und müssen in der vorgegebenen Reihenfolge stehen.
Schlüsselwortoperanden, z.B. PTERM=*ptermname*, können Sie in beliebiger Reihenfolge schreiben. Die Operanden müssen durch Kommas getrennt werden.

Wird ein optionaler Operand nicht gesetzt, dann gilt der Standardwert des Operanden.

Wird bei einem Kommando zur Modifikation der Konfiguration ein optionaler Operand nicht gesetzt, dann gilt weiterhin der durch die Generierung festgelegte Wert bzw. der zuvor durch die Administration gesetzte Wert.

Nach Bearbeitung eines Kommando liefert openUTM eine Ausgabe zurück, die das Resultat anzeigt. Dies bedeutet aber nicht in jedem Fall, dass die Aktion erfolgreich ausgeführt wurde.

Ob openUTM eine Aktion erfolgreich ausführen konnte, können Sie mit Hilfe der Informationsfunktionen überprüfen, z.B. mit dem Kommando KDCINF.

12.1 KDCAPPL - Eigenschaften und Grenzwerte für den Betrieb ändern

Mit KDCAPPL können Sie folgende Aktionen durchführen:

- Timereinstellungen und Maximalwerte ändern, die Sie in der KDCDEF-Steueranweisung MAX generiert haben.
- Die Anzahl der Prozesse (TASKS) festlegen, die sich gleichzeitig für die Anwendung einsetzen lassen. Wenn Sie die aktuelle Anzahl der Prozesse herabsetzen wollen, sollten Sie die Angaben im Abschnitt "Mögliche Maßnahmen" beachten.
- Die maximale Anzahl der Prozesse festlegen, die gleichzeitig asynchrone Vorgänge oder Vorgänge mit blockierenden Funktionsaufrufen (z.B. KDCS-Aufruf PGWT) bearbeiten dürfen.
  Die maximal mögliche Zahl dieser Prozesse ist abhängig von der Gesamtzahl der Prozesse der Anwendung und der in der KDCDEF-Anweisung MAX festgelegten Maximalzahl der Prozesse, die solche Vorgänge bearbeiten dürfen.
- Das Paging des Cache-Speichers steuern.
- Abrechnungs- und Kalkulationsphase für das UTM-Abrechnungsverfahren (Accounting) ein- und ausschalten.
- Die Datenkomprimierung ein- und ausschalten.
- Die Verbindung zu allen Druckern aufbauen, für die Nachrichten vorhanden sind.
- Das gesamte Anwendungsprogramm im laufenden Betrieb austauschen. Sie können damit, ohne die Anwendung zu beenden, Teilprogramme ändern und neue Teilprogramme, die Sie dynamisch in die Konfiguration aufgenommen haben, zum Anwendungsprogramm hinzufügen.

  In UTM-Anwendungen auf BS2000-Systemen werden damit auch Lademodule im Common Memory Pool getauscht, deren Version Sie zuvor mit KDCPROG geändert haben.
- Die System-Protokolldateien der Anwendung (SYSOUT und SYSLST bzw. stderr und stdout) im laufenden Betrieb umschalten. Sie verhindern damit einen Plattenengpass, weil Protokolldateien ausgewertet und nicht mehr benötigte Dateien gelöscht oder archiviert werden können.
- Darüber hinaus können Sie die Lieferung von Daten an den Software-Monitor openSM2 für die Anwendung ein- oder ausschalten.

Wirkungsdauer der Änderungen

Die mit KDCAPPL vorgenommenen Änderungen wirken höchstens für die Dauer des aktuellen Anwendungslaufs.

Ausnahme:
Ein Programmaustausch (Operand PROGRAM) wirkt über das Ende des aktuellen Anwendungslaufs hinaus. D.h. beim nächsten Start der Anwendung wird die zuletzt geladene Version des Anwendungsprogramms geladen. openUTM versucht bei einem Neu-Start das neue Anwendungsprogramm auch dann zu laden, wenn ein zuvor (in dem vorherigen Anwendungslauf) initiierter Programmaustausch nicht erfolgreich durchgeführt werden konnte.

Wirkung im Cluster

Die Wirkung in Cluster-Anwendungen ist bei den einzelnen Operanden beschrieben, da die mit KDCAPPL vorgenommenen Änderungen teils Knoten-lokal und teils Cluster-global wirken. Knoten-lokale Änderungen wirken höchstens für die Dauer des aktuellen Anwendungslaufs der Knoten-Anwendung.

Cluster-globale Änderungen wirken höchstens für die Dauer des aktuellen Anwendungslaufs der UTM-Cluster-Anwendung.

```
KDCAPPL     [ ,ACCOUNT={ ON | OFF } ]
            [  CACHE=%_utm_pages ]
            [ ,CALC={ ON | OFF } ]
            [ ,CONCTIME=con_control_time_sec ]
            [ ,CONRTIME=con_request_time_min ]
            [ ,DATA-COMPRESSION={ ON | OFF } ]
            [ ,MAXASYN=number_tasks ]
            [ ,MAX-CONN-USERS=number_users ]
            [ ,PGWTTIME=wait_time_sec ]
            [ ,PROGRAM={ NEW | OLD | SAME}  ]
            [ ,PTCTIME=wait_time_sec ]
            [ ,RESWAIT-PR=wait_time_sec ]
            [ ,RESWAIT-TA=wait_time_sec ]
            [ ,SPOOLOUT=ON ]
            [ ,SYSPROT=NEW ]
            [ ,TASKS=number_tasks ]
            [ ,TASKS-IN-PGWT=number_tasks ]
            [ ,TERMWAIT=wait_time_sec ]
            [ ,SM2={ ON | OFF } ]
```

Für die Administration über Message Queuing müssen Sie KDCAPPLA angeben.

ACCOUNT= schaltet die UTM-Abrechnungsphase ein bzw. aus.

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Cluster-global.

Auf BS2000-Systemen wird das Einschalten der Abrechnungsphase nur dann wirksam, wenn im BS2000-Accounting der Satztyp UTMA eingeschaltet ist. Das Kommando KDCAPPL ACC=ON wird nicht abgewiesen, wenn der Satztyp UTMA nicht eingeschaltet ist, da openUTM das erst beim Schreiben eines Abrechnungssatzes feststellen kann.

Beim Start der Anwendung gilt der bei der KDCDEF-Steueranweisung in ACCOUNT ACC= eingestellte Wert.

Zum UTM-Abrechnungsverfahren siehe auch das openUTM-Handbuch „Einsatz von UTM-Anwendungen“.

Hinweis für BS2000-Systeme:

- Mit KDCAPPL ACC=ON ist es auch möglich, nach einem Ausfall der BS2000-Abrechnung die Abrechnungsphase bei openUTM erneut einzuschalten, wenn die BS2000-Abrechnung wieder verfügbar ist.
- Beim RAV (**R**echenzentrum-**A**brechnungs-**V**erfahren) kann man bei der Auswertung der Accounting-Sätze festlegen, welche Tarife für welche Zeiträume verrechnet werden sollen.

CACHE=%_utm_pages

bestimmt, wieviel Prozent der Seiten im Cache-Speicher bei Engpasssituationen maximal auf KDCFILE gespeichert werden sollen.

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Cluster-global.

Mit CACHE können Sie den Prozentsatz, der bei der KDCDEF-Generierung in der MAX-Anweisung festgelegt wurde, für die Dauer der laufenden Anwendung anpassen. openUTM lagert bei einem Paging mindestens 8 UTM-Seiten aus, auch wenn sich ein niedrigerer Wert ergeben würde.

Minimalwert: 0 (in diesem Fall werden 8 Seiten ausgelagert)
Maximalwert: 100

> **i** Wurde bei der Generierung in der Steueranweisung MAX im Operanden CACHESIZE für *number* ein Wert < 100 angegeben, können bei der Ausgabe von *%_utm_pages* eines nachfolgenden KDCAPPL-Kommandos Rundungsfehler auftreten.

CALC= schaltet die Kalkulationsphase des UTM-Accounting ein oder aus.

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Cluster-global.

ON schaltet die Kalkulationsphase ein.

Auf BS2000-Systemen wird die Kalkulationsphase nur eingeschaltet, wenn im BS2000-Accounting der Accounting-Satztyp UTMK eingeschaltet ist.

OFF schaltet die Kalkulationsphase wieder aus.

Beim Start der Anwendung gilt der bei der KDCDEF-Steueranweisung in ACCOUNT ACC= eingestellte Wert.

Zum UTM-Accounting siehe auch das openUTM-Handbuch „Einsatz von UTM-Anwendungen“.

CONCTIME=con_control_time_sec

(**Con**nection **C**ontrol **Time**)
Zeit in Sekunden zur Überwachung des Aufbaus einer Session (LU6.1) bzw. Association (OSI TP). Wenn die Session bzw. Association innerhalb der angegebenen Zeit nicht aufgebaut wird, baut openUTM die Transportverbindung ab. Damit wird verhindert, dass eine Transportverbindung wegen eines misslungenen Aufbaus einer Session bzw. Association blockiert bleibt.

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Cluster-global.

CONCTIME=0 bedeutet:

- bei LU6.1-Sessions, dass der Aufbau nicht überwacht wird.
- bei OSI TP-Associations, dass maximal 60 Sekunden auf den Aufbau einer Association gewartet wird.

Maximalwert: 32767
Minimalwert: 0

CONRTIME=con_request_time_min

(**con**nection **r**equest **time**)
Zeit in Minuten, nach der openUTM beim Ausfall einer Verbindung zu einem Partner-Server bzw. zu einem Client oder Drucker versuchen soll, die Verbindung wieder aufzubauen.

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Cluster-global.

CONRTIME bezieht sich auf die Verbindungen zu folgenden Partnern:

- Drucker, zu denen openUTM beim Start der Anwendung automatisch eine Verbindung aufbaut, wenn diese Verbindung nicht zuvor durch die Administration abgebaut wurde.
- Drucker, zu denen openUTM eine Verbindung aufbaut, sobald die Anzahl der Druckaufträge für diesen Drucker den generierten Schwellwert überschreitet (LTERM-Partner mit *plev* > 0). openUTM versucht nur dann die Verbindung wieder aufzubauen, wenn die Anzahl der nach dem Verbindungsabbruch noch anstehenden Druckaufträge größer oder gleich dem Schwellwert ist.
  Ist in diesem Fall CONRTIME != 0, dann versucht openUTM auch dann die Verbindung zu dem Drucker aufzubauen, wenn die Verbindung zuvor explizit durch die Administration abgebaut wurde.
- TS-Anwendungen, die sich über LTERM-Partner an die UTM-Anwendung anschließen (eingetragen mit *ptype*='APPLI' oder 'SOCKET') und zu denen openUTM beim Start der Anwendung automatisch eine Verbindung aufbaut, wenn diese Verbindung nicht zuvor durch die Administration abgebaut wurde.
- Partner-Anwendungen, zu denen openUTM beim Start der Anwendung automatisch eine Verbindung aufbaut, wenn diese Verbindung nicht zuvor durch die Administration oder von openUTM wegen Ablauf eines Timers (*idletime*) abgebaut wurde.
- Nachrichtenverteiler (MUX) auf BS2000-Systemen, zu denen openUTM beim Start der Anwendung automatisch eine Verbindung aufbauen soll, wenn diese Verbindung nicht zuvor durch die Administration abgebaut wurde.

Kommt bei Partnern, die mit automatischem Verbindungsaufbau konfiguriert sind, keine Verbindung zustande (beim Start der Anwendung oder nach Verbindungsanforderung durch die Administration), dann versucht openUTM im Abstand von CONRTIME die Verbindung neu bzw. wieder aufzubauen.

Bei CONRTIME=0 unternimmt openUTM keinen Versuch, eine Verbindung wieder aufzubauen.

Maximalwert: 32767
Minimalwert: 0

DATA-COMPRESSION

schaltet die Datenkomprimierung ein oder aus. Eine Änderung wirkt über ein Anwendungsende hinaus.

Die pro Komprimierung eingesparten UTM-Seiten können per Kommando KDCINF STAT (siehe Abschnitt "Ausgabe von KDCINF (Beispiele)"), per Programmschnittstelle (siehe Abschnitt "kc_curr_par_str - Aktuelle Werte der Anwendungsparameter") oder per WinAdmin/WebAdmin abgefragt werden.

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Cluster-global.

ON schaltet die Datenkomprimierung ein.
Die Datenkomprimierung kann nur dann administrativ eingeschaltet werden, wenn die Datenkomprimierung per Generierung erlaubt ist (siehe openUTM-Handbuch „Anwendungen generieren“, KDCDEF-Anweisung MAX DATA-COMPRESSION=).

OFF schaltet die Datenkomprimierung aus.

MAXASYN=number_tasks

legt die maximale Anzahl der Prozesse der Anwendung fest, die gleichzeitig Aufträge an Asynchron-Transaktionscodes übernehmen dürfen (siehe KC_MODIFY_OBJECT, *obj_type* =KC_TASKS_PAR im Abschnitt "obj_type=KC_TASKS_PAR", Parameter *mod_max_asyntasks*).

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Knoten-lokal.

MAX-CONN-USERS=number_users

legt die maximale Anzahl der Benutzer fest, die gleichzeitig bei der UTM-Anwendung angemeldet sein dürfen. Durch diese Einschränkung können Sie verhindern, dass Ihre Anwendung überlastet wird.

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Knoten-lokal.

openUTM überprüft die Anzahl der aktiven Benutzer bei der Anmeldung eines weiteren Benutzers und lehnt die Anmeldung ab, wenn bereits *number_users* Benutzer angemeldet sind. Dies gilt nicht für Benutzerkennungen mit Administrationsberechtigung.

Sind in dem Moment, in dem Sie KDCAPPL aufrufen, mehr als *number_users* Benutzer angemeldet, wird kein Benutzer zwangsweise von der Anwendung abgemeldet. Es werden jedoch keine weiteren Anmeldungen zugelassen, bis die Anzahl der angeschlossenen Benutzer kleiner als *number_users* ist.

Ist eine Anwendung ohne Benutzerkennungen generiert, dann wird mit *number_users* die Anzahl der Dialog-Partner beschränkt, die gleichzeitig mit der Anwendung verbunden sein können. Wird für *number_users* eine Zahl angegeben, die größer ist als die Zahl der generierten Dialog-LTERM-Partner, dann hat *number_users* keine Wirkung. Dialog-LTERM-Partner sind alle LTERM-Partner, die mit USAGE=D generiert sind, LTERM-Partner der LTERM-Pools und - bei BS2000-Systemen - die LTERM-Partner, die openUTM intern für Multiplex-Anschlüsse erzeugt.

number_users=0 heißt, dass die Anzahl der angemeldeten Benutzer bzw. Dialog-Partner nicht beschränkt ist.

Maximalwert:

- BS2000-Systeme: 500000
- Unix-, Linux- und Windows-Systeme: Der Wert, der in der KDCDEF-Anweisung MAX CONN-USERS angegeben wurde

Minimalwert: 0 (keine Beschränkung).

PGWTTIME=wait_time_sec

ändert die bei der Generierung festgelegte Zeit in Sekunden, die ein Teilprogramm nach einem blockierenden Funktionsaufruf maximal auf das Eintreffen von Nachrichten warten darf, und zeigt den aktuell eingestellten Wert für die Wartezeit an. Die Wartezeit wird generiert in der KDCDEF-Anweisung MAX mit dem Operanden PGWTTIME.
Blockierende Funktionsaufrufe sind Aufrufe, bei denen die Kontrolle erst nach Eintreffen der Antwort an das Teilprogramm zurückgegeben wird (z.B. KDCS-Aufruf PGWT).

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Cluster-global.

Während dieser Wartezeit bleibt ein Prozess der UTM-Anwendung exklusiv für dieses Teilprogramm reserviert.

Maximalwert: 32767
Minimalwert: 60

PROGRAM=

Austausch des gesamten Anwendungsprogramms im laufenden Betrieb (siehe betreffendes openUTM-Handbuch „Einsatz von UTM-Anwendungen“, Programmaustausch).

Voraussetzungen für den Programmaustausch mit KDCAPPL PROGRAM=

- In dem neuen Anwendungsprogramm darf kein Teilprogramm fehlen, das vorher vorhanden war. Aufträge, die für einen Transaktionscode angenommen wurden, für den nach dem Programmaustausch kein Teilprogramm mehr da ist, werden von openUTM mit Fehler beendet (PEND ER).
- Einen Programmaustausch sollten Sie erst anstoßen, wenn alle zuvor eingegebenen Administrationsaufrufe von openUTM abgearbeitet worden sind. Insbesondere darf ein Programmaustausch nur angestoßen werden, wenn ein zuvor initiierter Programmaustausch vollständig abgearbeitet ist, d.h. der Programmaustausch für alle Prozesse der Anwendung durchgeführt ist.
- Auf BS2000-Systemen ist es Voraussetzung für einen Programmaustausch, dass die Anwendung mit mindestens einer LOAD-MODULE-Anweisung generiert ist.
- Damit Sie auf Unix-, Linux oder Windows-Systemen ein UTM-Anwendungsprogramm im laufenden Betrieb austauschen können, sollten die verschiedenen Versionen des Anwendungsprogramms (auch das aktuell geladene) mit Hilfe des UTM-Tools KDCPROG im Dateigenerationsverzeichnis PROG verwaltet werden. Das Dateigenerationsverzeichnis muss mit Hilfe von KDCPROG erstellt worden sein und im Basisverzeichnis *filebase* Ihrer Anwendung liegen.
  Der Programmaustausch ist ausführlich im openUTM-Handbuch „Einsatz von UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen“ beschrieben.

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Cluster-global für die aktuell laufenden Knoten-Anwendungen.

Zusammen mit PROGRAM sind weitere Operanden von KDCAPPL wirkungslos.
KDCAPPL PROGRAM= wird abgewiesen, wenn zum Zeitpunkt des Aufrufs gerade ein Programmaustausch durchgeführt wird.

Auf BS2000-Systemen sind nur die Werte NEW und SAME erlaubt. Beide Werte haben die gleiche Wirkung.

NEW Das gesamte Anwendungsprogramm soll ausgetauscht werden.

BS2000-Systeme

KDCAPPL PROGRAM=NEW ist nicht erlaubt, wenn die Anwendung im Dialog gestartet ist.

In einer UTM-Anwendung unter einem BS2000-System wird das Anwendungsprogramm in allen Prozessen entladen und anschließend neu geladen. Dabei werden die aktuellen Versionen der Lademodule geladen.
Für Lademodule, die mit VERSION=*HIGHEST-EXISTING generiert sind, wird die höchste in der Bibliothek vorhandene Version geladen.

Anwendungsteile, die in Common Memory Pools liegen, werden dabei ebenfalls ausgetauscht. Die Version der Lademodule, die bei dem Programmaustausch geladen werden sollen, müssen Sie zuvor mit KDCPROG bekannt machen.

Vor dem Programmaustausch können mehrere Lademodule im Common Memory Pool durch mehrere KDCPROG-Aufrufe für den Austausch vorgemerkt werden.

Das Beenden des Anwendungsprogramms mit anschließendem Neustart bewirkt außerdem, dass alle Lademodule, die mit Lademodus ONCALL generiert sind, entladen werden.

Statische Anwendungsteile lassen sich damit auch austauschen, wenn man die Anwendung zuvor neu bindet.

Unix-, Linux- und Windows-Systeme

In einer UTM-Anwendung auf Unix-, Linux- und Windows-Systemen lädt openUTM das Anwendungsprogramm aus der nächsthöheren Dateigeneration, die im Dateigenerationsverzeichnis *filebase*/PROG (Unix- und Linux-Systeme) bzw. *filebase*\PROG (Windows-Systeme) steht, wenn die verschiedenen Versionen des Anwendungsprogramms (auch das aktuell geladene) mit Hilfe des UTM-Tools KDCPROG im Dateigenerationsverzeichnis PROG verwaltet werden.
Falls Sie kein Dateigenerationsverzeichnis erstellt haben, dann lädt KDCAPPL PROG=NEW das Anwendungsprogramm (*utmwork)* neu, das im Basisverzeichnis *filebase* liegt.

SAME Auf Unix-, Linux- und Windows-Systemen lädt openUTM das Anwendungsprogramm aus derselben Dateigeneration, die im Dateigenerationsverzeichnis *filebase*/PROG (Unix- und Linux-Systeme) bzw. *filebase*\PROG (Windows-Systeme) steht, wenn die verschiedenen Versionen des Anwendungsprogramms (auch das aktuell geladene) mit Hilfe des UTM-Tools KDCPROG im Dateigenerationsverzeichnis PROG verwaltet werden.

Falls Sie kein Dateigenerationsverzeichnis erstellt haben, dann lädt KDCAPPL PROG=SAME das Anwendungsprogramm (*utmwork*) neu, das im Basisverzeichnis *filebase* liegt.

Auf BS2000-Systemen wirkt SAME wie NEW.

OLD (nur auf Unix-, Linux- und Wndows-Systemen)

openUTM lädt das Anwendungsprogramm aus der nächstniedrigeren Dateigeneration, die im Dateigenerationsverzeichnis *filebase*/PROG (Unix- und Linux-Systeme) bzw. *filebase*\PROG (Windows-Systeme) steht, wenn die verschiedenen Versionen des Anwendungsprogramms (auch

das aktuell geladene) mit Hilfe des UTM-Tools KDCPROG im Dateigenerationsverzeichnis PROG verwaltet werden.

Damit können Sie z.B. auf das ursprüngliche Anwendungsprogramm zurückschalten, wenn sich im Betrieb mit dem neuen Anwendungsprogramm Fehler zeigen.

Falls Sie kein Dateigenerationsverzeichnis erstellt haben, dann lädt KDCAPPL PROG=OLD das Anwendungsprogramm (*utmwork)* neu, das im Basisverzeichnis *filebase* liegt.

Einen erneuten Programmaustausch akzeptiert openUTM erst, wenn der Austausch für alle Prozesse abgeschlossen ist.

Treten beim Start des neuen Anwendungsprogramms für den ersten Prozess Fehler auf, dann gibt openUTM die Meldung K075 aus und lädt wieder das ursprüngliche Anwendungsprogramm.

Ist der Austausch des Anwendungsprogramms für alle Prozesse abgeschlossen, gibt openUTM die Meldung K074 aus.

Auf Unix-, Linux- und Windows-Systemen können Sie die Generation des aktuell geladenen Anwendungsprogramms z.B. mit KDCINF SYSP abfragen.

PTCTIME=wait_time_sec

nur für Anwendungen mit verteilter Verarbeitung:
wait_time_sec ist die Zeit in Sekunden, die ein lokaler Auftragnehmer-Vorgang maximal im Zustand PTC (prepare to commit, Transaktionsstatus P) auf eine Quittung vom Auftraggeber-Vorgang wartet. Nach Ablauf der Zeit wird die Verbindung zum Auftraggeber abgebaut, die Transaktion im lokalen Auftragnehmer-Vorgang zurückgesetzt und der Vorgang beendet. Dabei kann es evtl. zu einem Mismatch kommen.

Wurde für die Anwendung bereits KDCSHUT WARN oder GRACE gegeben und der Wert von PTCTIME ist ungleich 0, so wird die Wartezeit unabhängig von PTCTIME so gewählt, dass die Transaktion zurückgesetzt wird, bevor die Anwendung beendet wird, um eine abnormale Anwendungsbeendigung mit ENDPET möglichst zu vermeiden.

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Cluster-global.

Der Wert 0 bedeutet, dass beliebig lange Zeit auf eine Quittung gewartet wird.

Maximalwert: 32767
Minimalwert: 0

RESWAIT-PR=wait_time_sec

Zeit in Sekunden, die maximal auf ein von einem anderen Prozess gesperrtes Betriebsmittel gewartet werden soll. Wird diese Zeit überschritten, dann beendet sich die Anwendung mit einer Fehlermeldung.

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Cluster-global.

Bei der Angabe von RESWAIT-PR ist zu beachten, dass der Wert von *wait_time_sec* mindestens so groß sein muss wie die längste Bearbeitungszeit (Realzeit) für die folgenden Fälle:

- Bei Transportsystem-Anwendungen, die keine SOCKET-Anwendungen sind (Clients mit PTYPE=APPLI), sind die Betriebsmittel für die Dauer eines Verarbeitungsschrittes inklusive VORGANG-Exit bei Vorgangsbeginn und/oder Vorgangsende gesperrt.

- Bei Vorgangsende sind die Betriebsmittel gesperrt, solange das VORGANG-Exit-Programm läuft.

Maximalwert: 32767
Minimalwert: 300

RESWAIT-TA=wait_time_sec

Zeit in Sekunden, die ein Teilprogramm maximal auf ein von einer anderen Transaktion gesperrtes Betriebsmittel warten soll: GSSBs, ULSs, TLSs. Ist nach Ablauf dieser Zeit das Betriebsmittel nicht verfügbar, erhält das Anwendungsprogramm einen entsprechenden Returncode KCRCCC.

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Cluster-global.

Die in *wait_time_sec* angegebene Wartezeit hat keine Bedeutung, wenn die Sperre von einer Mehrschritt-Transaktion gehalten wird, die auf eine Eingabe-Nachricht nach einem PEND KP oder einem PGWT KP wartet. In diesem Fall erhalten alle Teilprogramme, die auf die gesperrten Betriebsmittel zugreifen, sofort (ohne Wartezeit *wait_time_sec*) einen Returncode KCRCCC.

RESWAIT-TA=0 bedeutet, dass nicht gewartet wird. Ein Teilprogrammlauf, der auf gesperrte Betriebsmittel zugreifen will, erhält sofort einen Returncode KCRCCC.

Die reale Wartezeit ist abhängig von der Genauigkeit, mit der die Börsenwartezeiten im Betriebssystem eingestellt sind.

Maximalwert: 32767
Minimalwert: 0

SPOOLOUT=ON

bewirkt, dass openUTM zu allen Druckern eine Verbindung aufbaut, für die zum Aufrufzeitpunkt Nachrichten vorliegen und die noch nicht mit der Anwendung verbunden sind.
Damit können Sie alle Nachrichten an die Drucker ausgeben, zu denen eine Verbindung aufgebaut werden kann (z.B. vor einer Beendigung der Anwendung).

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Knoten-lokal.

SYSPROT=NEW

Die Protokolldateien der Anwendung werden umgeschaltet.

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Cluster-global für die aktuell laufenden Knoten-Anwendungen.

Die Namen der neuen Protokolldateien werden wie folgt gebildet:

BS2000-Systeme:

SYSOUT: *präfix*.O.*YYMMDD*.*HHMMSS*.*TSN*
SYSLST: *präfix*.L.*YYMMDD*.*HHMMSS*.*TSN*

Wenn die Anwendung im Dialog gestartet ist, wird nur SYSLST umgeschaltet.

präfix Präfix, das Sie beim Starten der UTM-Anwendung für den Startparameter SYSPROT angegeben haben (siehe openUTM-Handbuch „Einsatz von UTM-Anwendungen auf BS2000-Systemen“).

Standardwert in stand-alone UTM-Anwendungen:
Name der Anwendung, der bei der KDCDEF-Generierung in MAX APPLINAME festgelegt wurde.

Standardwert in UTM-Cluster-Anwendungen:
Name der Anwendung, der bei der KDCDEF-Generierung in MAX APPLINAME festgelegt wurde, gefolgt von einem Punkt und dem Rechnernamen aus der CLUSTER-NODE-Anweisung für den eigenen Knoten.

YYMMDD

Datum des Umschaltzeitpunkts

HHMMSS

Uhrzeit des Umschaltzeitpunkts

TSN TSN der Task

Unix-, Linux- und Windows-Systeme:

stdout: *präfix*`.out.`*YY-MM-DD*`.`*HHMMSS*
stderr: *präfix*`.err.`*YY-MM-DD*`.`*HHMMSS*

präfix Präfix, das Sie beim Starten der UTM-Anwendung für den Startparameter SYSPROT angegeben haben (siehe openUTM-Handbuch „Einsatz von UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen“).

Standardwert: utmp

YY-MM-DD

Datum des Umschaltzeitpunkts

HHMMSS

Uhrzeit des Umschaltzeitpunkts

TASKS=number_tasks

legt die aktuelle Anzahl der Prozesse der Anwendung fest. openUTM schaltet Prozesse entsprechend zu oder ab (siehe KC_MODIFY_OBJECT, *obj_type*=KC_TASKS_PAR im Abschnitt "obj_type=KC_TASKS_PAR", Parameter *mod_max_tasks*).

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Knoten-lokal.

Maximalwert:
der bei der Generierung festgelegte Maximalwert für TASKS (KDCDEF-Steueranweisung MAX ..., TASKS=...)

Minimalwert:

- Wenn MAX TASKS-IN-PGWT=0: 1

- Wenn MAX TASKS-IN-PGWT>0:
  TASKS WAITING IN PGWT +1, aber mindestens 2
  (TASKS WAITING IN PGWT kann mit KDCINF SYSP abgefragt werden).

Das Starten und Beenden der UTM-Prozesse benötigt eine gewisse Zeit. Sie sollten deshalb nach der Eingabe von KDCAPPL TASKS= zunächst das Ergebnis des Aufrufs abwarten und mit KDCINF SYSPARM überprüfen, bevor Sie erneut KDCAPPL TASKS= aufrufen. Andernfalls können Startfehler auftreten.

TASKS-IN-PGWT=number_tasks

legt die Anzahl der Prozesse der UTM-Anwendung fest, die gleichzeitig Teilprogramme mit blockierenden Aufrufen (z.B. KDCS-Aufruf PGWT) bearbeiten dürfen (siehe KC_MODIFY_OBJECT, *obj_type*=KC_TASKS_PAR im Abschnitt "obj_type=KC_TASKS_PAR", Parameter *mod_max_tasks_in_pgwt*).

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Knoten-lokal.

Das Kommando wird abgewiesen, wenn Sie TASKS-IN-PGWT=0 angeben, obwohl bei der Generierung für MAX TASKS-IN-PGWT >0 generiert wurde.

TERMWAIT=wait_time_sec

Zeit in Sekunden, die bei einer Mehrschritt-Transaktion (d.h. im Programm PEND KP) maximal zwischen einer Ausgabe an einen Dialog-Partner (Terminal, UPIC-Client, TS-Anwendung oder Auftraggeber-Vorgang einer Partner-Anwendung) und der nachfolgenden Dialog-Antwort verstreichen darf. Wird die Zeit in *wait_time_sec* überschritten, dann wird die Transaktion zurückgesetzt. Die Verbindung zum Dialog-Partner wird abgebaut.

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Cluster-global.

Maximalwert: 32767
Minimalwert: 60

SM2=

schaltet die Datenlieferung an openSM2 ein bzw. aus. Die Lieferung von Daten an openSM2 ist nur möglich, wenn das Einschalten der openSM2-Messung per Generierung erlaubt wurde (KDCDEF-Anweisung MAX, Operand SM2=ON oder OFF). Wurde SM2=NO generiert, so kann die Datenlieferung an openSM2 durch die Administration nicht eingeschaltet werden.

In UTM-Cluster-Anwendungen gilt:
Der Operand wirkt Cluster-global. Wird eine Knoten-Anwendung mit neuer Generierung gestartet, so gilt der Wert aus der neuen Generierung für diese Knoten-Anwendung sowie für alle weiteren neu startenden Knoten-Anwendungen

ON openUTM soll Daten für openSM2 bereitstellen.

OFF Die Bereitstellung von Daten an openSM2 wird ausgeschaltet.

Ausgabe von KDCAPPL

Es werden (mit Ausnahme von KDCAPPL PROG=) die neuen und die alten Werte der Parameter angezeigt.

```
                      NEW     OLD
TERMWAIT              ...     ...
RESWAIT-PR            ...     ...
RESWAIT-TA            ...     ...
CONRTIME              ...     ...
CURR TASKS            ...     ...
MAXASYN               ...     ...
TASKS-IN-PGWT         ...     ...
CACHE                 ...     ...
ACCOUNT               ...     ...
CALC                  ...     ...
SM2                   ...     ...
MAX-CONN-USERS        ...     ...
PGWTTIME              ...     ...
CONCTIME              ...     ... (1.)
PTCTIME               ...     ... (1.)
PROGRAM FGG           ...     ... (2.)
```

1. Die Zeilen für PTCTIME und CONCTIME werden nur bei einer Anwendung mit verteilter Verarbeitung über LU6.1 oder OSI TP ausgegeben.
2. Die Zeile für PROGRAM FGG wird nur ausgegeben, wenn in einer UTM-Anwendung auf Unix-, Linux- und Windows-Systemen das gesamte Anwendungsprogramm mit KDCPROG ausgetauscht werden soll. In diesem Fall wird für NEW die neue und für OLD die alte Generationsnummer des Anwendungsprogramms ausgegeben. Ist der Programmaustausch für alle Prozesse durchgeführt, gibt openUTM die Meldung K074 aus.

12.2 KDCBNDL - Master-LTERMs austauschen

Mit KDCBNDL können Sie die Master-LTERMs zweier LTERM-Bündel austauschen (siehe openUTM-Handbuch „Anwendungen generieren“). Alle Slave-LTERMs und die zugehörigen PTERMs des ersten LTERM-Bündels werden dem zweiten Master-LTERM zugeordnet und umgekehrt.

Dieses Kommando ist nur in stand-alone UTM-Anwendungen erlaubt.

Wirkungsdauer der Änderung

Die Änderung bleibt über das Ende der Anwendung hinaus erhalten.

```
KDCBNDL    master-lterm1, master-lterm2
```

Für die Administration über Message Queuing müssen Sie KDCBNDLA angeben.

master-lterm1 Name des ersten Master-LTERM

master-lterm2 Name des zweiten Master-LTERM

Ausgabe von KDCBNDL

Am Administrator-Terminal werden folgende Meldungen angezeigt:

- Wenn das Kommando erfolgreich durchgeführt wurde:
  `COMMAND ACCEPTED - 'master-lterm1' AND 'master-lterm2' SWITCHED`
- Wenn das Kommando nicht durchgeführt werden konnte:
  `COMMAND REJECTED`
- Wenn für *master-lterm1* oder *master-lterm2* ein LTERM angegeben wurde, das kein Master-LTERM ist:
  `COMMAND REJECTED - 'lterm' NO MASTER-LTERM`

12.3 KDCDIAG - Diagnosehilfen ein- und ausschalten

Mit KDCDIAG können Sie UTM-Funktionen ein- und ausschalten, die Sie bei der Diagnose von Fehlern unterstützen. Folgende Funktionen können angefordert werden:

- Testmodus ein- oder ausschalten
- im laufenden Betrieb einen UTM-Dump zur Diagnose erzeugen
- openUTM veranlassen, bei Eintritt eines bestimmten Ereignisses einen UTM-Dump zu erstellen
- UTM-Messmonitor KDCMON ein- und ausschalten
- die BCAM-Tracefunktion ein- und ausschalten, die alle Verbindungs-bezogenen Aktivitäten innerhalb der Anwendung verfolgt. Die BCAM-Tracefunktion kann auch nur für Verbindungen zu einzelnen, explizit angegebenen Kommunikationspartnern eingeschaltet werden.
- die OSS-Tracefunktion ein- und ausschalten. Der OSS-Trace hilft bei der Diagnose von Problemen mit OSI TP-Verbindungen.
- Debug-Informationen für den Anschluss an die Datenbank ausgeben
- Protokolldateien für die UTM-Anwendung umschalten

Wirkung im Cluster

Die Wirkung in UTM-Cluster-Anwendungen ist bei den einzelnen Operanden beschrieben, da die mit KDCDIAG vorgenommenen Änderungen teils Knoten-lokal und teils Cluster-global wirken. Knoten-lokale Änderungen wirken höchstens für die Dauer des aktuellen Anwendungslaufs. Cluster-globale Änderungen wirken höchstens für die Dauer des aktuellen Cluster-Anwendungslaufs.

Wirkungsdauer der Änderungen

Jede Änderung wirkt für die Dauer des Anwendungslaufs oder bis sie zurückgesetzt wird.

```
KDCDIAG     [  DUMP=YES ]

           [ ,{ DUMP-MESSAGE | DUMP-MESSAGE{1|2|3} }=
                  { (MSG, msg-nr) | (SIGN, sign) | (RCCC, rccc) |
                    (RCDC, rcdc)  | *NONE } ]
           [ ,INSERT1= (insert-nr, value, { EQ | NE }) ]

           [ ,INSERT2= (insert-nr, value, { EQ | NE }) ]

           [ ,INSERT3= (insert-nr, value, { EQ | NE }) ]

           [ ,KDCMON=  { ON | OFF } ]

           [ ,TESTMODE={ ON | OFF } ]

           [ ,BTRACE=  { ON | OFF }  ,

                  { LTERM ={ ltermname | (ltermname_1,...,ltermname_10) } |
                    LPAP  ={ lpapname  | (lpapname_1,...,lpapname_10)   } |
                    USER  ={ username  | (username_1,...,username_10)   } |
                    MUX   =( mux-name, proname, bcamappl ) [1]
                  } ]

           [ ,OTRACE= { ON | (SPI,INT,OSS,SERV,PROT) | OFF } ]

           [ ,STXIT-LOG={ ON | OFF } ] [1]

           [ ,XA-DEBUG={ YES | ALL | OFF} ]

           [ ,XA-DEBUG-OUT={ SYSOUT | FILE} ]
```

[1] nur auf BS2000-Systemen

Für die Administration über Message Queuing müssen Sie KDCDIAGA angeben.

DUMP=YES

fordert im laufenden Betrieb einen UTM-Dump an. Der UTM-Dump wird (nur von einem Prozess der Anwendung) mit dem Dump-Grund "REASON=DIAGDP" erzeugt.

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Knoten-lokal.

DUMP-MESSAGE[1...3]=

Sie spezifizieren hier ein Ereignis, bei dessen Eintreten openUTM einen UTM-Dump erzeugen soll. Das Kommando KDCDIAG DUMP-MESSAGE= wird nur ausgewertet, wenn der Testmodus eingeschaltet ist (TESTMODE=ON).

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Cluster-global.

Der Dump wird von dem Prozess erstellt, bei dem das Ereignis aufgetreten ist. Die Anwendung wird nicht beendet.

Mit den Parametern DUMP-MESSAGE1, DUMP-MESSAGE2 und DUMP-MESSAGE3 können Sie bis zu drei unterschiedliche Ereignisse festlegen, bei deren Auftreten ein Message-Dump gezogen werden soll. Die Angabe von DUMP-MESSAGE ist dabei synonym zu DUMP-MESSAGE1.

Pro Kommando KDCDIAG können Sie maximal einen Parameter DUMP-MESSAGE[i] angeben.

Folgende Ereignisse können Sie spezifizieren:

- die Ausgabe einer Meldungsnummer der Form K*nnn* oder P*nnn*,
- das Auftreten eines bestimmten KDCS-Returncodes (CC oder DC),
- das Auftreten eines bestimmten SIGNON-Statuscode.

Das Kennzeichen des Dumps ist abhängig vom Ereignis:

- Bei K- oder P-Meldungen hat es das Präfix „ME“, gefolgt von der Meldungsnummer, z.B. MEP012.
- Bei einem primären KDCS-Returncode hat es das Präfix „CC-“ gefolgt vom Returncode, z.B. CC-71Z.
- Bei einem sekundären KDCS-Returncode hat es das Präfix „DC“ gefolgt vom Returncode, z.B. DCK303.
- Bei einem SIGNON-Statuscode hat es das Präfix „SG-“ gefolgt vom Status, z.B. SG-U01.

MSG, msg-nr

Für *msg-nr* geben Sie eine UTM-Meldungsnummer in der Form K*nnn* oder P*nnn* an. Bei jedem Auftreten der spezifizierten Meldungsnummer wird ein Dump erzeugt, bis die Meldungsnummer zurückgesetzt wird.

Ausnahme sind hier die Meldungsnummern K023, K043, K061 oder K062, bei denen nur einmal ein Dump erzeugt und dann der Message-Dump automatisch ausgeschaltet wird.

SIGN, sign

Für *sign* geben Sie einen SIGNON-Statuscode an (dreistellig), z.B. U04, wobei KCRSIGN1 den Wert U,I,A oder R haben muss. Beim Auftreten dieses Codes beim Anmelden eines Benutzers wird ein UTM-Dump mit dem Kennzeichen SG-U04 von dem Prozess erzeugt, bei dem der SIGNON-Status aufgetreten ist. Das passiert unabhängig davon, ob in der Anwendung ein Anmelde-Vorgang generiert ist oder nicht. Anschließend wird der Message-Dump für dieses Ereignis automatisch ausgeschaltet.

RCCC, rccc
RCDC, rcdc

Für *rccc* geben Sie einen KDCS-Returncode (KCRCCC, z.B. „40Z“) an, für *rcdc* einen inkompatiblen KDCS-Returncode (KCRCDC, z.B. „KD10“). Beim Auftreten dieses Returncodes bei einem KDCS-Aufruf wird ein UTM-Dump mit dem Kennzeichen CC-40Z oder DCKD10 von dem Prozess erzeugt, in dem der Returncode aufgetreten ist. Anschließend wird der Message-Dump für dieses Ereignis automatisch ausgeschaltet.

Bei allen KDCS-Returncodes >= 70Z und den zugehörigen inkompatiblen KDCS-Returncodes, bei denen grundsätzlich kein PENDER-Dump geschrieben wird (z.B. 70Z /K316), wird auch kein Dump erzeugt.

*NONE Explizites Ausschalten eines Ereignisses für einen Message-DUMP.

INSERT1...INSERT3=(insert-nr, value, {EQ | NE})

Sie können hier für die Meldung *msg-nr* bis zu drei Inserts als zusätzliche Bedingungen angeben, um das Erstellen eines Dump weiter einzuschränken. INSERTx wird nur ausgewertet, wenn auch DUMP-MESSAGE[i] angegeben ist.

Ein Message-Dump wird nur gezogen, wenn alle in INSERT1 ... INSERT3 spezifizierten Kriterien erfüllt sind.

Die Reihenfolge der Inserts einer Meldung finden Sie im jeweiligen systemspezifischen openUTM-Handbuch „Meldungen, Test und Diagnose“.

insert-nr
bezeichnet die Nummer des Inserts, das geprüft werden soll, z.B. „2“ für das zweite Insert einer Meldung.

value
gibt den Wert des Inserts an, gegen den geprüft werden soll. Folgende Angaben sind möglich:

- nnn: numerisch, Wertebereich $0...2^{31}-1$
- [C]'aaa': alphanumerisch, maximale Länge 32 Bytes
- X'xxx': hexadezimal, maximale Länge 32 Bytes

EQ | NE
legt fest, ob auf Gleichheit oder Ungleichheit geprüft werden soll.
Standard: EQ

KDCMON= schaltet den UTM-Messmonitor ein bzw. aus.

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Knoten-lokal.

ON schaltet den UTM-Messmonitor ein.
Der Messmonitor KDCMON ist im jeweiligen openUTM-Handbuch „Einsatz von UTM-Anwendungen“ beschrieben.

OFF schaltet den UTM-Messmonitor wieder aus.

TESTMODE= schaltet den Testmodus ein bzw. aus.

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Cluster-global.

ON Der Testmodus wird eingeschaltet, das bedeutet, es laufen zusätzliche UTM-interne Routinen zur Plausibilitätsprüfung ab und es werden interne TRACE-Informationen aufgezeichnet. Trace-Informationen werden im KTA und - bei OSI TP-Anwendungen - auch im XAPTP-Baustein geschrieben.
Der Testmodus sollte nur zum Erstellen von Diagnoseunterlagen eingeschaltet werden.

OFF — Testmodus ausschalten.

Standard: Anzeigen der aktuellen Einstellung.

BTRACE= — Ein-/Ausschalten der BCAM-Tracefunktion von UTM. Der BCAM-Trace von UTM verfolgt alle Verbindungs-spezifischen Aktivitäten innerhalb einer UTM-Anwendung.
Beim Einschalten der Tracefunktion, im Folgenden BTRACE-Funktion genannt, erzeugt jeder Prozess der Anwendung seine eigene Trace-Datei, in der er die Verbindungs-spezifischen Ereignisse aufzeichnet.
Wird die BTRACE-Funktion ausgeschaltet, werden die Trace-Dateien geschlossen und können danach ausgewertet werden. Inhalt und Auswertung der Trace-Datei sind im openUTM-Handbuch „Meldungen, Test und Diagnose“ beschrieben.

Die BTRACE-Funktion kann bereits beim Start der Anwendung über den Startparameter BTRACE eingeschaltet werden.

Sie können die BTRACE-Funktion allgemein für alle Verbindungen der Anwendung oder Partner-spezifisch für die Verbindungen zu bestimmten LTERM- und LPAP-Partnern einschalten.

ON — Die BTRACE-Funktion wird allgemein eingeschaltet, es werden die Ereignisse bzgl. aller Verbindungen zu jedem beliebigen Kommunikationspartner der Anwendung (Clients und Partner-Anwendung der verteilten Verarbeitung über LU6.1) protokolliert.

In UTM-Cluster-Anwendungen gilt: BTRACE=ON ohne weitere Parameter wirkt Cluster-global.

ON, LPAP=lpapname|(lpapname_1,...,lpapname_10)

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Knoten-lokal.

oder

ON, LTERM=ltermname|(ltermname_1,...,ltermname_10)

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Knoten-lokal.

oder

ON, USER=username|(username_1,...,username_10)

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Cluster-global.

oder

ON, MUX=(mux_name,proname,bcamappl) (nur auf BS2000-Systemen)

Die BTRACE-Funktion wird Partner-spezifisch eingeschaltet, es werden alle Ereignisse bzgl. der Verbindungen zu den angegebenen Partnern oder Benutzern bzw. zu den MUX-Partnern protokolliert.

Anzugeben sind:

- Für *lpapname_...* die Namen von LPAP-Partnern
- Für *ltermname_...* die Namen von LTERM-Partnern, die Clients zugeordnet sind

- Für *username_...* die Namen von Benutzern, deren Ereignisse unabhängig von der genutzten Verbindung aufgezeichnet bzw. nicht aufgezeichnet werden sollen. Dies ist insbesondere bei der Nutzung von TPOOLs hilfreich.
- Nur auf BS2000-Systemen: Bei MUX der Name und der Prozessor des MUX-Partners sowie der Transportsystemzugangspunkt, über den sich der MUX-Partner mit der Anwendung verbindet.

Die BTRACE-Funktion kann nur dann explizit für Verbindungen zu bestimmten Partner-Anwendungen, Clients oder Benutzern eingeschaltet werden, wenn sie nicht bereits für alle Verbindungen der Anwendung eingeschaltet ist.

Wollen Sie die BTRACE-Funktionen für einige Partner-Anwendungen und für einige Clients einschalten, dann rufen Sie das Kommando KDCDIAG mehrmals auf:

```
KDCDIAG BTRACE=ON, LPAP=...
KDCDIAG BTRACE=ON, LTERM=...
KDCDIAG BTRACE=ON, USER=...
KDCDIAG BTRACE=ON, MUX=... (nur auf BS2000-Systemen)
```

OFF

Die BTRACE-Funktion wird für alle Verbindungen der Anwendung ausgeschaltet, auch wenn sie Partner-spezifisch eingeschaltet wurde.

In UTM-Cluster-Anwendungen gilt: BTRACE=OFF ohne weitere Parameter wirkt Cluster-global.

OFF, LPAP=lpapname|(lpapname_1,...,lpapname_10)

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Knoten-lokal.

oder

OFF, LTERM=ltermname|(ltermname_1,...,ltermname_10)

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Knoten-lokal.

oder

OFF, USER=username|(username_1,...,username_10)

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Cluster-global.

oder

OFF, MUX=(mux_name,proname,bcamappl) (nur auf BS2000-Systemen)

Die BTRACE-Funktion wird für Verbindungen zu den in *lpapname_1,...,lpapname_10* angegebenen Partner-Anwendungen bzw. zu den in *ltermname_1,...,ltermname_10* angegebenen Clients bzw. zu den in *username_1,...,username_10* angegebenen Benutzern bzw. dem MUX-Partner ausgeschaltet.
Die BTRACE-Funktion kann nur Partner-spezifisch ausgeschaltet werden, wenn sie für die Verbindungen zu diesen Partnern auch explizit eingeschaltet wurde (mit BTRACE=ON, LPAP=... bzw. LTERM=... bzw. MUX=...)

OTRACE= Ein-/Ausschalten der Tracefunktion von OSS.
Der OSS-Trace wird zur Diagnose bei Problemen mit OSI TP-Verbindungen der Anwendung benötigt.

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Cluster-global.

Die Tracefunktion von OSS kann auch beim Start der Anwendung über den Startparameter [.UTM] START ... OTRACE= ein- bzw. ausgeschaltet werden.

Es werden die Trace-Records der Typen SPI, INT, OSS, SERV und PROT protokolliert.

ON Schaltet die OSS-Tracefunktion für alle Record-Typen ein.
Beim Einschalten der OSS-Tracefunktion erzeugt jeder Prozess der Anwendung seine eigene Trace-Datei.

(SPI, INT, OSS, SERV, PROT)

Schaltet die OSS-Tracefunktion ein. Es werden die Trace-Records des angegebenen Typs protokolliert. Die Reihenfolge der Typangabe ist beliebig.

SPI
Das XAP-TP System Programming Interface wird protokolliert.

INT
Der interne Ablauf im XAP-TP-Baustein wird protokolliert.

OSS
Die OSS-Aufrufe werden protokolliert.

SERV
Die OSS-internen Trace-Records vom Typ =O_TR_SERV werden protokolliert.

PROT
Die OSS-internen Trace-Records vom Typ =O_TR_PROT werden protokolliert.

OFF Schaltet die OSS-Tracefunktion aus, die Trace-Dateien werden geschlossen und können ausgewertet werden. Siehe dazu auch das openUTM-Handbuch „Meldungen, Test und Diagnose“ und das Handbuch zu OSS.

STXIT-LOG= Nur auf BS2000-Systemen: Ein/Ausschalten des erweiterten STXIT-Loggings bei Problemen mit der STXIT-Behandlung. Bei Eintreten eines STXIT-Ereignisses werden mehrere K099-Meldungen auf SYSOUT ausgegeben.

ON schaltet das STXIT-Logging ein.

OFF schaltet das STXIT-Logging aus.

XA-DEBUG= gibt an, ob Debug-Informationen für den Anschluss an eine XA-Datenbank ausgegeben werden sollen.

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Knoten-lokal.

YES XA-DEBUG wird eingeschaltet, die Aufrufe an die XA-Schnittstelle werden protokolliert.

ALL Erweitertes XA-DEBUG, zusätzlich zu den Aufrufen an die XA-Schnittstelle werden bestimmte Datenbereiche ausgegeben.

OFF XA-DEBUG wird ausgeschaltet.

XA-DEBUG-OUT=

steuert die Ausgabe-Ziele von XA-DEBUG.

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Knoten-lokal.

SYSOUT Die Protokollierung erfolgt auf SYSOUT/stderr, Standardwert.

FILE Die Protokollierung erfolgt in eine Datei.

Wenn Sie im Kommando KDCDIAG nur XA-DEBUG angeben, ohne XA-DEBUG-OUT zu versorgen, so wird ggf. der Wert verwendet, den Sie beim Starten der UTM-Anwendung im Startparameter angegeben haben (siehe openUTM-Handbuch „Einsatz von UTM-Anwendungen“). Andernfalls wird auf SYSOUT/stderr, protokolliert.

i Eine Modifikation der beiden Operanden XA-DEBUG und XA-DEBUG-OUT ist nur in einer UTM-Anwendung sinnvoll, in der ein Datenbank-Anschluss über das XA-Interface generiert wurde.

Ausgabe von KDCDIAG

Bei KDCDIAG DUMP=YES wird die Meldung „DIAGNOSTIC DUMP CREATED“ ausgegeben.

Bei den anderen Operanden zeigt openUTM am Administrator-Terminal die neuen und alten Einstellungen für die Diagnosehilfen an:

```
STATUS          NEW                         OLD
TESTMODE        ON|OFF                      ON|OFF
KDCMON          ON|OFF                      ON|OFF
OSS-TRACE       ON|OFF                      ON|OFF
                SPI INT OSS SERV PROT       SPI INT OSS SERV PROT
BTRACE          ON S|ON A|OFF               ON S|ON A|OFF
LTERM/LPAP/USER BTRACE
                NEW                         OLD
ltermname       ON|OFF                      ON|OFF
lpapname        ON|OFF                      ON|OFF

username        ON|OFF                      ON|OFF
STXIT-LOG       ON|OFF                      ON|OFF
XA-DEBUG        YES|ALL|OFF                 YES|ALL|OFF
XA-DEBUG-OUT    SYSOUT|FILE                 SYSOUT|FILE
```

Erläuterungen zur Ausgabe

TESTMODE Die Zeile für TESTMODE wird immer ausgegeben, unabhängig davon, ob der KDCDIAG-Aufruf den Operanden TESTMODE enthält oder nicht.

KDCMON Die Zeile für KDCMON wird immer ausgegeben, unabhängig davon, ob der KDCDIAG-Aufruf den Operanden KDCMON enthält oder nicht.

BTRACE

Die Zeile für BTRACE wird immer ausgegeben. Bei eingeschaltetem BTRACE (ON) wird zusätzlich ausgegeben, ob die Tracefunktion für alle Verbindungen der Anwendung (ON A; A=all) oder nur für Verbindungen zu einigen Kommunikationspartnern (ON S; S=select) eingeschaltet ist.

OSS-TRACE Die Zeile für OSS-TRACE wird immer ausgegeben, wenn im KDCDIAG-Aufruf der Operand OTRACE angegeben wurde

LTERM/LPAP/USER

Wird nur ausgegeben, wenn die BCAM-Tracefunktion explizit für Verbindungen zu bestimmten Kommunikationspartnern (LPAP-, LTERM- oder MUX-Partner) oder Benutzern eingeschaltet wird/war. Es wird hier für die einzelnen Kommunikationspartner bzw. Benutzer, für die die BCAM-Tracefunktion eingeschaltet ist/war, der aktuelle und alte BTRACE-Zustand ausgegeben.

12.4 KDCHELP - Syntax der Administrationskommandos abfragen

Mit KDCHELP können Sie sich über die Syntax der Administrationskommandos informieren.

```
KDCHELP      [command]
```

Für die Administration über Message Queuing müssen Sie KDCHELPA angeben.

command Für *command* geben Sie den Namen des Administrationskommandos an, für das openUTM die Syntax ausgeben soll.

openUTM liefert die Namen aller Dialog-Kommandos von KDCADM zusammen mit einer kurzen Funktionsbeschreibung der einzelnen Kommandos zurück, wenn Sie Folgendes eingeben:

`KDCHELP` (d.h. keine Angabe für *command*)
oder
`KDCHELP KDCHELP` (d.h. Angabe von KDCHELP für *command*)
oder
`KDCHELP XXX` (*XXX* = ungültiger Name)

Gültige Angaben für *command* sind:

KDCAPPL
KDCBNDL
KDCDIAG
KDCHELP
KDCINF
KDCLOG
KDCLPAP
KDCLSES
KDCLTAC
KDCLTERM
KDCPOOL
KDCPROG
KDCPTERM
KDCSHUT
KDCSLOG
KDCSWTCH
KDCTAC
KDCTCL
KDCUSER
Für UTM-Anwendungen auf BS2000-Systemen können Sie darüber hinaus noch folgende Namen für *command* angeben:
KDCMUX
KDCSEND

12.5 KDCINF - Informieren über Objekte und Anwendungsparameter

Mit KDCINF können Sie Namen und Eigenschaften der Objekte der Anwendung, generierte Anwendungsparameter und Statistikwerte über die Auslastung der Anwendung abfragen. Im Parameter *type* legen Sie fest, worüber Sie sich informieren wollen.

Wirkung in UTM-Cluster-Anwendungen:

Die ausgegebenen Informationen beziehen sich immer nur auf die lokale Knoten-Anwendung, an der der Auftrag ausgeführt wurde.

Umfang der Informationsausgabe einschränken

Mit einem KDCINF-Aufruf können Sie die Eigenschaften von Objekten eines bestimmten Typs abfragen. Mit den Operanden CONT, LIST und PRONAM bestimmen Sie den Umfang der Informationen, die openUTM ausgeben soll. Sie können explizit angeben, über welche Objekte openUTM informieren soll, und den Umfang der Informationen pro Objekt bestimmen:

- In LIST können Sie explizit die Namen der Objekte angeben, über die openUTM informieren soll.
- Durch Angabe von LIST=KDCNAMES können Sie die Ausgabe auf eine Namensliste aller Objekte des Typs einschränken. Andere Eigenschaften werden nicht ausgegeben.
- LIST=KDCCON bewirkt, dass openUTM nur die Eigenschaften aller derzeit aktiven Objekte des Typs anzeigt, d. h. Eigenschaften von Clients, Druckern oder Partner-Servern, zu denen eine Verbindung besteht oder von Benutzern, die derzeit angemeldet sind.
- Mit CONT steuern Sie, mit welchem Objekt die Ausgabeliste beginnen soll. Die Listen sind alphabetisch nach den Namen der Objekte geordnet. In CONT geben Sie einen Namen an. Das kann ein beliebiger Name sein, es muss nicht der Name eines existierenden Objektes sein. Die Ausgabeliste beginnt mit diesem Objekt, wenn der angegebene Name der Name eines Objektes ist. Existiert kein Objekt mit dem in CONT angegebenen Namen, dann beginnt die Liste mit dem alphabetisch folgenden Objekt. Informationen zu den Objekten, deren Namen in der alphabetischen Reihenfolge vor dem in CONT angegebenen Namen liegen, werden nicht ausgegeben.
- Mit PRONAM können Sie die Ausgabe von Objekteigenschaften und -namen auf Objekte beschränken, die sich auf einem bestimmten Rechner befinden.

Eine Einschränkung der Ausgaben von KDCINF ist in vielen Fällen sinnvoll, z.B. bei großen Anwendungen und bei der Ausgabe der Informationen am Terminal. Eine Gesamtausgabe aller Informationen zu einem Objekttyp ist oft so umfangreich, dass sie bei der Ausgabe am

Administrator-Terminal viele Bildschirmseiten umfasst. Die Übersichtlichkeit geht dann verloren. Bei großen Anwendungen nimmt die Erstellung von Gesamtlisten durch openUTM sehr viel Zeit in Anspruch.

Aus diesem Grund sollten Sie bei größeren Anwendungen keine Gesamtlisten über Objekte eines bestimmten Typs bzw. aller Objekte und Anwendungsparameter anfordern, also Abfragen folgender Form vermeiden:

```
KDCINF ..,LIST=KDCALL,OUT=KDCPRINT oder
KDCINF ..,LIST=KDCALL,OUT=KDCBOTH
```

Ausgabe der Informationen

Mit dem Operanden OUT bestimmen Sie, wo openUTM die angeforderten Informationen ausgibt. Sie können sich Informationen direkt am Administrator-Terminal anzeigen lassen, sich die Informationen auf einem Drucker ausgeben lassen oder die Informationen auch direkt an ein Teilprogramm (Asynchron-TAC) übergeben, das die gelieferten Informationen weiter verarbeitet.

Für einige Objekte wie z.B. TAC ist in der Ausgabezeile von KDCINF nicht ausreichend Platz, um alle numerischen Werte in ihrer maximalen Länge darstellen zu können, beispielsweise den Wert des Feldes IN-Q. Werden diese Werte zu groß für die Ausgabe mit KDCINF, dann werden diese Werte sinnvoll verkürzt und in Gleitpunktdarstellung angezeigt. D.h. es werden die höchstwertigen Stellen gefolgt von einem Exponenten e ausgegeben. Der ungefähre tatsächliche Wert des Feldes ergibt sich dann aus den führenden Ziffern multipliziert mit 10 „hoch“ e.

Beispiel:

> Bei der Ausgabe von KDCINF TAC stehen für das Feld IN-Q vier Stellen zur Verfügung. Ist die Anzahl der Nachrichten, die von dem TAC noch bearbeitet werden müssen, größer als 9.999, dann kommt die verkürzte Darstellung zur Anwendung. Ein Wert von 11.235 wird dargestellt als 11e3, d.h. der tatsächliche Wert liegt im Bereich zwischen 11.000 und 11.999.

Berechnung der Mittelwerte

Die mit KDCINF angezeigten Mittelwerte werden für die ersten 32767 Werte als arithmetisches Mittel berechnet. Danach wird der neue Wert jeweils mit 1/32767 gewichtet und der bisherige Mittelwert mit 32767/32768. Es ist möglich, dass bei der Berechnung der Mittelwerte Ungenauigkeiten durch Rundungsfehler auftreten. Das ist insbesondere dann der Fall, wenn der neue Wert sich erheblich vom Mittelwert unterscheidet.

Beispiele für die Ausgaben und eine Erläuterung der Ausgabeinformation finden Sie im Anschluss an die Beschreibung der Operanden (siehe Abschnitt "Ausgabe von KDCINF (Beispiele)").

Besonderheiten bei der Ausgabe am Terminal:
Passt eine angeforderte Ausgabe nicht auf einen Bildschirm, so bietet openUTM i.A. in der letzten Bildschirm-Zeile ein Fortsetzungskommando an, mit dem die Ausgabe an der aktuellen Position fortgesetzt werden kann.

Wenn Sie in der Liste mit Fortsetzungskommando weiterblättern wollen, so müssen Sie:

- an BS2000-Systemen nur ein Zeichen in diesem vorgegebenen Kommando überschreiben und die <DUE>-Taste drücken.
- an Unix-, Linux- und Windows-Systemen das angebotene Fortsetzungskommando eingeben.

12.5.1 KDCINF Syntax-Beschreibung

```
KDCINF      type
            [ ,LIST={ KDCNAMES | (name_1,...,name_10) | KDCALL | KDCCON } ]
            [ ,OUT={ KDCDISP | KDCPRINT | KDCBOTH | ltermname | tacname } ]
            [ ,CONT={ name | (name,proname) | (name,proname,bcamname) } ]
            [ ,PRONAM=proname]
            [ ,LPAP=lpapname ]                    (nur bei type = LSES)
            [ ,OSI-LPAP=osilpapname ]             (nur bei type = OSI-ASSOCIATIONS)
           Nur auf BS2000-Systemen:
            [ ,OPTION=MONITORING ]                (nur bei type = MUX)
```

Für die Administration über Message Queuing müssen Sie KDCINFA angeben.

type Typ der Objekte bzw. Anwendungsparameter, über die openUTM informieren soll. Für *type* können Sie einen der folgenden Werte angeben:

ALL **KS**ET **LT**ERM PAGEPOOL POOL	PROG **PT**ERM **STAT**ISTICS **S**YS**LOG** **SYSP**ARM	TAC TACCLASS **T**AC-**P**ROG **U**SER
Zur Abfrage von Informationen über die Objekte der verteilten Verarbeitung können Sie für *type* folgende Werte angeben:		
CON LPAP LSES LTAC	**OSI-ASS**OCIATIONS OSI-CON OSI-LPAP	
In UTM-Anwendungen auf BS2000-Systemen können Sie für *type* auch angeben:		
LOAD-**MOD**ULE	MUX	
In UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen können Sie für *type* auch angeben:		
SHARED-OBJECT		

Die Angaben für *type* haben die folgende Bedeutung:

ALL fordert eine Gesamtinformation über alle Objekte, Statistikwerte und Anwendungsparameter an.

Das Ergebnis von KDCINF ALL wird immer auf dem Standard-System-Drucker (im Betriebsystem voreingestellter Drucker) ausgegeben.
Die Steuerung der Ausgabe über den Operanden OUT ist **nicht** möglich, Angaben für OUT

werden ignoriert.
Für den Operanden LIST werden nur die Operandenwerte KDCNAMES (Standard) und KDCALL berücksichtigt.

Bei der Kombination ALL und LIST=KDCNAMES werden die Namen aller Objekte ausgegeben, jedoch keine Anwendungsparameter und keine Statistikinformationen.

Bei der Kombination ALL und LIST=KDCALL werden alle Anwendungsparameter, Statistikwerte und die Eigenschaften aller Objekte ausgegeben.

In UTM-Anwendungen auf BS2000-Systemen wird bei KDCINF ALL, LIST=KDCALL keine Information über Lademodule ausgegeben.

Bei KDCINF ALL bleiben Angaben für die Operanden CONT, OUT, PRONAM ohne Wirkung.

KSET Informieren über die Keysets der Anwendung. Die Information über ein bestimmtes Keyset (Verwendung von LIST=*kset_name*) können Sie sich am Administrator-Terminal ausgeben lassen. Wollen Sie sich über mehrere oder alle Keysets informieren, dann erfolgt die Ausgabe immer am Standard-System-Drucker. Angaben für den Operanden OUT bleiben ohne Wirkung.

Ausnahme

Wurde bei der KDCDEF-Generierung in der MAX-Anweisung für den Operanden KEYVALUE ein Wert größer als 255 angegeben, dann können Sie mit KDCINF **keine** Informationen über Keysets (*type* = KSET) abfragen.
Das KDCINF-Kommando wird in diesem Fall mit der Meldung „KEYVALUE > 255 NOT SUPPORTED“ abgewiesen.
Sie können sich jedoch mit Hilfe der Programmschnittstelle zur Administration ein eigenes Administrationsprogramm zur Abfrage der Information erstellen (KC_GET_OBJECT-Aufruf mit Objekttyp KC_KSET).

LOAD-MODULE (nur auf BS2000-Systemen)

Informieren über Lademodule. Der Umfang der Ausgabe kann mit den Operanden CONT und LIST gesteuert werden. Bei LIST ist nur die Angabe von KDCNAMES oder eines einzelnen Lademodul-Namens erlaubt. Bei LIST=KDCNAMES wird eine Liste aller Lademodul-Namen ausgegeben. Informationen über ein bestimmtes Lademodul erhalten Sie, wenn Sie für LIST den Namen des Lademoduls angeben.

Geben Sie in LIST den Namen eines Lademoduls an, dann wird die Angabe in CONT als Programmname interpretiert. Die Angabe in CONT bestimmt dann, mit welchem Teilprogrammnamen die Liste der im Lademodul enthaltenen Teilprogramme beginnt.

LTERM Informieren über LTERM-Partner, d.h. über logische Namen und Eigenschaften von Clients und Druckern. Der Umfang der Ausgabe kann mit den Operanden CONT und LIST gesteuert werden.

Ist einem LTERM-Partner ein Druckerbündel (mehrere Drucker, d.h. PTERMs) zugeordnet, dann können Sie sich mit folgendem Kommando die Liste der zum LTERM gehörenden Drucker (PTERMs) anzeigen lassen.

```
KDCINF LTERM, LIST=ltermname
```

Ist das angegebene LTERM das Primary-LTERM einer LTERM-Gruppe, so werden die Primary- und Gruppen-LTERMs der LTERM-Gruppe ausgegeben (siehe openUTM-Handbuch „Anwendungen generieren“). Die Reihenfolge ist wie folgt:

- Die erste Zeile enthält das Primary-LTERM.
- Die Folgezeilen enthalten die Gruppen-LTERMs.

Ist das angebene LTERM sowohl MASTER-LTERM eines LTERM-Bündels als auch Primary-LTERM seiner LTERM-Gruppe, so werden alle Master-, Primary-, Slave-und Gruppen-LTERMs ausgegeben. Die Reihenfolge ist wie folgt:

- Die erste Zeile enthält das Master-/primary-LTERM
- Dann folgen alle Gruppen-LTERMs
- Die Folgezeilen enthalten alle Slave-LTERMS. Als erstes steht hier das Slave-LTERM, an das die Nachrichten zugestellt werden.

Der Aufruf KDCINF LTERM, LIST=*master-lterm* gibt die Master- und Slave-LTERMs eines LTERM-Bündels aus. Die Ausgabe erfolgt genauso wie beim Aufruf KDCINF LTERM, LIST=KDCALL:

- Die erste Zeile enthält das Master-LTERM.
- Die Folgezeilen enthalten die Slave-LTERMs.

MUX (nur auf BS2000-Systemen)

Informieren über die Eigenschaften und den aktuellen Zustand von Multiplex-Anschlüssen.
Wird MUX zusammen mit dem Operanden OPTION=MONITORING angegeben, dann liefert openUTM zusätzlich Messwerte für die Multiplex-Verbindungen.
OPTION=MONITORING ist jedoch ohne Wirkung, wenn Sie LIST=KDCNAMES angeben.

PAGEPOOL Informieren über die aktuelle Belegung des Pagepools.
Zusammen mit PAGEPOOL hat nur der Operand OUT eine Wirkung.
Angaben für die Operanden LIST, CONT und PRONAM werden von openUTM ignoriert.

POOL Informieren über LTERM-Pools. Der Umfang der Ausgabe kann mit den Operanden CONT und LIST gesteuert werden.

PROG ist nur zulässig, wenn die Anwendung mit Lademodulen/SharedObjects generiert ist (LOAD-MODULE-Anweisung (BS2000-Systeme) und SHARED-OBJECT-Anweisung (Unix-, Linux- und Windows-Systeme)).
openUTM informiert über die Teilprogramme der Anwendung.
Zu jedem Teilprogramm wird der Name des zugehörigen Lademoduls/Shared Objects/DLLs, dessen Lademodus und eine Aussage zu dessen Austauschbarkeit angezeigt.
Der Umfang der Ausgabe kann mit den Operanden CONT und LIST gesteuert werden.

PTERM Informieren über physische Eigenschaften von Clients und Druckern. Der Umfang der Ausgabe kann mit den Operanden CONT, LIST und PRONAM gesteuert werden.

SHARED-OBJECT (nur auf Unix-, Linux- und Windows-Systemen)

ist nur zulässig, wenn die Anwendung mit SHARED-OBJECT-Anweisungen generiert ist.
openUTM informiert über Shared Objects/DLLs.
Der Umfang der Ausgabe kann mit den Operanden CONT und LIST gesteuert werden. Bei LIST ist nur die Angabe von KDCNAMES oder eines einzelnen Shared Object/DLL-Namens erlaubt.
Bei LIST=KDCNAMES wird eine Liste aller Shared Object/DLL-Namen ausgegeben.

STATISTICS Anzeigen allgemeiner Statistik-Informationen.
Zusammen mit STATISTICS hat nur der Operand OUT eine Wirkung.
Angaben für die Operanden LIST, CONT und PRONAM werden von openUTM ignoriert.

Einige Statistikdaten werden über die K081-Meldung stündlich auf die System-Protokolldatei SYSLOG geschrieben und nach dem Schreiben wieder auf 0 gesetzt. Dazu muss die Anwendung mit MAX STATISTICS-MSG=FULL-HOUR generiert sein. Im Abschnitt "Ausgabe von KDCINF (Beispiele)" sind die Statistikwerte, die openUTM liefert, und deren Lebensdauer beschrieben.

SYSLOG Informieren über die SYSLOG-Datei der UTM-Anwendung. Zusammen mit SYSLOG hat nur der Operand OUT eine Wirkung. Angaben für die Operanden LIST, CONT und PRONAM werden von openUTM ignoriert.
KDCINF SYSLOG wirkt wie KDCSLOG INFO (siehe Abschnitt "KDCSLOG - SYSLOG-Datei administrieren").

SYSPARM Informieren über Anwendungsparameter (Systemparameter) und Timereinstellungen, die bei der Generierung in der MAX-Anweisung festgelegt wurden und per Administration geändert werden können. Mit SYSPARM können Sie z.B. Parameterwerte kontrollieren, die mit KDCAPPL verändert wurden.
Zusammen mit SYSPARM hat nur der Operand OUT eine Wirkung.
Angaben für die Operanden LIST, CONT und PRONAM werden von openUTM ignoriert.

TAC Informieren über Transaktionscodes bzw. TAC-Queues der Anwendung.
Der Umfang der Ausgabe kann mit Hilfe der Operanden CONT und LIST gesteuert werden.

TAC-PROG ist nur zulässig, wenn die Anwendung mit Lademodulen/Shared Objects generiert ist (LOAD-MODULE-Anweisung (BS2000-Systeme) bzw. SHARED-OBJECT-Anweisung (Unix-, Linux- und Windows-Systeme)). openUTM informiert darüber, welche Teilprogramme den Transaktionscodes zugeordnet sind und zu welchen Lademodulen/Shared Objects/DLLs die Teilprogramme gehören. Die Transaktionscodes werden im Operanden LIST angegeben.

TACCLASS Informieren über TAC-Klassen der Anwendung.
openUTM zeigt an, wieviele Nachrichten in jeder TAC-Klasse auf Bearbeitung warten, wie hoch die durchschnittliche Wartezeit pro TAC-Klasse ist und ob die Prioritätensteuerung für die TAC-Klassen generiert ist.
Ist die Prioritätensteuerung nicht generiert, d.h. bei der KDCDEF-Generierung wurde die Anweisung TAC-PRIORITIES nicht angegeben, dann wird angezeigt, wieviele Prozesse jeder TAC-Klasse zugeordnet sind.

USER Informieren über die Benutzerkennungen der Anwendung.
openUTM informiert über Sicherheitsverletzungen für die Benutzerkennung, verbrauchte CPU-Zeit seit dem Anmelden und den LTERM-Partner, über den die Benutzerkennung angemeldet ist.

Der Umfang der Ausgabe kann mit Hilfe der Operanden CONT und LIST gesteuert werden.

Die folgenden Werte sind nur für Anwendungen mit verteilter Verarbeitung sinnvoll:

CON Nur bei verteilter Verarbeitung über das LU6.1-Protokoll.
openUTM informiert über Verbindungen, die mit KC_CREATE_OBJECT Objekttyp KC_CON erzeugt oder mit der KDCDEF-Steueranweisung CON generiert wurden. openUTM zeigt Namen, generierte Eigenschaften, aktuellen Zustand und Statistikwerte über die Auslastung der Verbindung

an.
Der Umfang der Ausgabe kann mit Hilfe der Operanden CONT, LIST und PRONAM gesteuert werden.

LPAP
Nur bei verteilter Verarbeitung über das LU6.1-Protokoll.
openUTM informiert über Namen und Eigenschaften der LPAP-Partner.
Abhängig von den Angaben in LIST gibt openUTM entweder nur die Namen oder die Namen zusammen mit den Eigenschaften der LPAP-Partner aus.
Der Umfang der Ausgabe kann mit Hilfe der Operanden CONT und LIST gesteuert werden.

Mit dem Aufruf KDCINF LPAP, LIST=*master-lpap* können Sie sich die Master- und Slave-LPAPs eines LU6.1-LPAP-Bündels ausgeben lassen (siehe openUTM-Handbuch „Anwendungen generieren “). Die Ausgabe erfolgt genauso wie beim Aufruf KDCINF LPAP, LIST=KDCALL:

- Die erste Zeile enthält das Master-LPAP.
- Die Folgezeilen enthalten die Slave-LPAPs.

LSES
Nur bei verteilter Verarbeitung über das LU6.1-Protokoll.
openUTM informiert über lokale Sessions, die mit KC_CREATE_OBJECT Objekttyp KC_LSES erzeugt oder mit der KDCDEF-Steueranweisung LSES generiert wurden. Geben Sie LSES zusammen mit dem Operanden LPAP=*lpapname* an (KDCINF LSES, LPAP=*lpapname*), dann schränkt openUTM die Ausgabe auf Informationen über Sessions ein, die für den in *lpapname* angegebenen LPAP-Partner generiert sind.
Der Umfang der Ausgabe kann auch mit Hilfe der Operanden CONT und LIST gesteuert werden.

LTAC
Informieren über Namen und Eigenschaften, die fernen Service-Programmen innerhalb der lokalen Anwendung zugeordnet sind (LTAC-Eigenschaften).
Der Umfang der Ausgabe kann mit Hilfe der Operanden CONT und LIST gesteuert werden.

OSI-CON
Nur bei verteilter Verarbeitung über das OSI TP-Protokoll.
openUTM informiert über die mit der KDCDEF-Steueranweisung OSI-CON generierten Namen für logische Verbindungen zu Partner-Anwendungen.
Abhängig von den Angaben in LIST werden die in der Anweisung OSI-CON generierten Eigenschaften der zugehörigen Verbindung angezeigt.

Der Umfang der Ausgabe kann mit Hilfe der Operanden CONT und LIST gesteuert werden.

OSI-LPAP
Nur bei verteilter Verarbeitung über das OSI TP-Protokoll.
openUTM informiert über OSI-LPAP-Partner, die für die OSI TP-Partner-Anwendungen der lokalen Anwendung generiert wurden. Abhängig von den Angaben für den Operanden LIST gibt openUTM entweder nur die Namen oder die Namen zusammen mit den logischen Eigenschaften der Partner-Anwendung aus.
Der Umfang der Ausgabe kann mit Hilfe der Operanden CONT und LIST gesteuert werden.
Der Aufruf KDCINF OSI-LPAP, LIST=*master-lpap-name* gibt die Master- und Slave-LPAPs eines OSI-LPAP-Bündels aus (siehe openUTM-Handbuch „Anwendungen generieren“). Die Ausgabe erfolgt genauso wie beim Aufruf KDCINF OSI-LPAP, LIST=KDCALL:

- Die erste Zeile enthält das Master-OSI-LPAP.
- Die Folgezeilen enthalten die Slave-OSI-LPAPs.

OSI-ASSOCIATIONS

Nur bei verteilter Verarbeitung über das OSI TP-Protokoll.
openUTM informiert über OSI TP-Associations. Es werden Informationen zum Auftraggeber, der eine Association belegt, und Statistikinformationen angezeigt.

KDCINF...,L=KDCNAMES
gibt die Namen der generierten OSI-Associations aus.

KDCINF...,L=KDCALL,OSI-LPAP=osilpapname
gibt nur die aktuell verbundenen OSI-Associations aus, sortiert nach der von XAPTP vergebenen Association-Id.
Der Operand OSI-LPAP=*osilpapname* ist Pflicht!

KDCINF...,L=(*name_1...name_10),*OSI-LPAP=osilpapname
In diesem Fall muss bei *name_n* die von XAPTP vergebene Association-Id angegeben werden, nicht der generierte OSI-Association Name.
Der Operand OSI-LPAP=*osilpapname* ist Pflicht!

Der Umfang der Ausgabe kann mit Hilfe der Operanden CONT und LIST gesteuert werden.

openUTM schränkt die Ausgabe von Informationen auf die OSI-Associations ein, die zu dem angegebenen OSI-LPAP-Partner aufgebaut sind.

Die folgenden Operanden steuern die Ausgabe

LPAP=lpapname

ist nur bei *type*=LSES zulässig.
Der Operand schränkt die Ausgabe der Session-Eigenschaften auf die Sessions ein, die für die in *lpapname* angegebene Partner-Anwendung generiert wurden.

OPTION=MONITORING (nur auf BS2000-Systemen)

ist nur bei *type*=MUX zulässig und wirkt nur bei LIST != KDCNAMES.
openUTM informiert über Messwerte von Multiplex-Verbindungen.

Bei KDCINF ALL werden diese Messwerte nicht mit ausgegeben.

CONT= Fortsetzen/Beginnen der Ausgabeliste an einer bestimmten Stelle. Die Ausgabe der Listen ist alphabetisch nach Objektnamen geordnet, CONT=*name* bewirkt, dass die ausgegebene Liste erst mit dem Objekt *name* beginnt und nur die Objekte enthält, deren Name in der alphabetischen Reihenfolge hinter *name* liegen.

Bei UTM-Anwendungen auf BS2000-Systemen hat der Operand CONT zusammen mit LIST=*name* nur dann eine Wirkung, wenn für *type* LOAD-MODULE und für *name* der Name eines Teilprogramms angegeben wird.

In UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen hat der Operand CONT, wenn er zusammen mit LIST=(*name_1*,..., *name_10*) angegeben wird, keine Wirkung.

name Die Liste beginnt mit dem Objekt *name.* Für *name* ist der Name eines Objektes der Anwendung anzugeben. Folgende Namen können Sie angeben:

- bei *type*=KSET: KSET-Name eines Keysets

- bei *type*=LTERM: Logischer Name eines Client /Druckers (Name eines LTERM-Partners)
- bei *type*=PTERM: (PTERM-)Name eines Client oder Druckers
- bei *type*=POOL: das für einen LTERM-Pool definierte LTERM-Präfix
- bei *type*=PROG: Name eines Teilprogramms
- bei *type*=TAC: TAC-Name eines lokalen Transaktionscodes/Queue
- bei *type*=USER: Benutzerkennung (USER-Name)
- bei *type*=CON/OSI-CON: ein in einer CON- bzw. OSI-CON-Anweisung generierter Name einer logischen Verbindung
- bei *type*=LPAP/OSI-LPAP: Name eines LPAP- oder OSI-LPAP-Partners
- bei *type*=LTAC: lokaler TAC-Name eines fernen Service-Programms
- bei *type*=OSI-ASSOCIATION: Association-Id, die der Association beim Aufbau einer OSI TP-Verbindung zugeordnet wurde

In UTM-Anwendungen auf BS2000-Systemen können Sie zusätzlich die folgenden Namen angeben:

- bei *type*=LOAD-MODULE: Name eines Lademoduls oder eines Teilprogramms
- bei *type*=MUX: Name eines Multiplexanschlusses

(name,proname)

Die Liste soll mit dem Objekt *(name,proname)* beginnen. *proname* ist der Name des Prozessors, an dem sich das Objekt *name* befindet. Die Angabe von *proname* ist nur sinnvoll, wenn Objekte mit *type*=PTERM / CON / MUX mit demselben Namen existieren und deshalb eine eindeutige Identifizierung nur über den unterschiedlichen Prozessornamen möglich ist.

(name,proname,bcamappl)

Die Liste soll mit dem Objekt (*name*,*proname*,*bcamappl*) beginnen. *bcamappl* ist der Name des Transportzugriffspunkts, über den sich das Objekt (*name*,*proname*) an die Anwendung anschließt. Die Angabe von *bcamappl* ist nur sinnvoll, wenn Objekte mit *type*=PTERM / CON / MUX mit demselben Namen und Prozessor existieren und deshalb eine eindeutige Identifizierung nur über den unterschiedlichen Namen des Transportzugriffspunkts möglich ist.

Die Ausgabe beginnt bei dem Objekt *(name,proname)*, dem der in *bcamappl* angegebene Name des lokalen Transportzugriffspunkts zugeordnet ist.

LIST= steuert Art und Umfang der Informationen.

KDCNAMES

Es wird eine Namensliste aller Objekte des in *type* angegebenen Objektyps ausgegeben. LIST=KDCNAMES ist ohne Wirkung bei *type*=PAGEPOOL, STATISTICS, SYSLOG, SYSPARM, TACCLASS.

(name_1,..., name_10)

Es werden die Eigenschaften der Objekte mit den Namen *name_1,..., name_10* angezeigt. Sie können maximal 10 Namen angeben, bei einem Namen können die Klammern entfallen.

KDCCON Nur sinnvoll bei *type*=PTERM, USER, LSES und CON.

Für UTM-Anwendungen auf BS2000-Systemen auch bei *type*=MUX.

Es werden nur die Eigenschaften der Objekte angezeigt, die zur Zeit mit der Anwendung verbunden sind.

Ausnahmen bei type=USER:
Ist die Anwendung mit SIGNON MULTI-SIGNON=NO generiert, dann werden Benutzerkennungen, über die nur OSI TP-Partner zum Starten von Asynchron-Vorgängen angemeldet sind, nicht angezeigt.

Ist die Anwendung mit SIGNON MULTI-SIGNON=YES generiert, dann werden folgende Benutzerkennungen nicht angezeigt:

- Benutzerkennungen mit RESTART=NO, die nicht über ein Terminal angemeldet sind
- Benutzerkennungen, über die nur OSI TP-Partner angemeldet sind, die entweder die Functional Unit „COMMIT“ ausgewählt haben, oder die einen Asynchron-Vorgang starten.

KDCALL Die Eigenschaften aller Objekte des in *type* angegebenen Typs werden angezeigt.

Standard: KDCNAMES

OSI-LPAP=osilpapname

ist nur bei *type*=OSI-ASSOCIATIONS zulässig. Der Operand schränkt die Ausgabe von Informationen auf die OSI-Associations ein, die zu dem angegebenen OSI-LPAP-Partner aufgebaut sind.

Die Angabe des Operanden ist Pflicht bei:
KDCINF OSI-ASSOCIATION...,L=(*name_1...name_10)*
KDCINF OSI-ASSOCIATION...,L=KDCALL

OUT= gibt an, wohin openUTM die angeforderten Informationen ausgeben soll.

KDCDISP Ausgabe am Administrator-Terminal, d.h. an dem Terminal, an dem KDCINF eingegeben wurde.

KDCPRINT Auf Unix- und Linux-Systemen wird für die Ausgabe das Shell-Script `admlp` verwendet. `admlp` ist in `$UTMPATH/shsc` enthalten und ruft das Kommando `lp` auf. Der Anwender kann `admlp` modifizieren oder ein eigenes Script namens `admlp` erstellen und unter einem eigenen Verzeichnis ablegen. Dieses Verzeichnis muss dann in der Pfadvariablen `$PATH` (vor `$UTMPATH/shsc`) enthalten sein.

Auf Windows-Systemen wird die Ausgabe auf Drucker aus der UTM-Anwendung heraus derzeit noch nicht unterstützt, d. h. es wird keine Datei erzeugt und auch nicht gedruckt.

KDCBOTH Ausgabe am Administrator-Terminal und (auf Unix- und Linux-Systemen) auf Standard-System-Drucker.

Auf Unix- und Linux-Systemen wird für die Ausgabe das Shell-Script `admlp` (s.o.) verwendet.

ltermname Die Ausgabe erfolgt auf dem Drucker mit dem logischen Namen *ltermname*.

tacname

Name des Transaktionscodes, an den openUTM das Ergebnis der Informationsabfrage übergeben soll. Der Transaktionscode muss einem Teilprogramm zugeordnet sein, das in einem Asynchron-Vorgang abläuft.

Standard: KDCDISP

PRONAM=proname

wirkt nur bei *type*=PTERM, CON und MUX.
openUTM liefert nur Informationen über die Clients und Partner-Anwendungen, die auf dem Rechner *proname* ablaufen bzw. am Rechner *proname* angeschlossen sind.

Standardwert für openUTM auf Unix- und Linux-Systemen: Leerzeichen für lokale Geräte

12.5.2 Ausgabe von KDCINF

Es werden die Ausgaben in Abhängigkeit von *type* aufgelistet. Dargestellt werden die Ausgaben aller Eigenschaften (LIST!=KDCNAMES).
Bei Eingabe von KDCINF ALL,LIST=KDCALL werden mit Ausnahme der Informationen über Lademodule, und Shared Objects und DLLs alle im Folgenden beschriebenen Einzelausgaben hintereinander ausgegeben.

type=CON

Die Ausgabe ist abhängig davon, ob einem CON-Objekt ein kurzer oder ein langer Rechnername zugeordnet ist. Bei einem langen Rechnernamen wird die Information zu einem CON-Objekt in zwei Bildschirmzeilen ausgegeben.

```
CON  PRONAM    LPAP  BCAMAPPL  STA     CONNECT    CTIME     LETTERS   CONB
con  proname   lpap  applname  ON|OFF  Y|N|W  A   minutes   number    number
CON  PRONAM    LPAP  BCAMAPPL  STA     CONNECT    CTIME     LETTERS   CONB
con  long.processor.name
               lpap  applname  ON|OFF  Y|N|W  A   minutes   number    number
```

Erläuterungen zur Ausgabe

CON	Der mit KC_CREATE_OBJECT Objekttyp KC_CON erzeugt oder mit der KDCDEF-Steueranweisung CON generierte Name für die logische Verbindungen zur Partner-Anwendung *lpap*.
PRONAM	Name des Rechners, auf dem die Partner-Anwendung abläuft.
LPAP	Logischer Name der Partner-Anwendung, für die die logische Verbindung generiert ist.
BCAMAPPL	Name der lokalen UTM-Anwendung (BCAMAPPL-Name), über den die Verbindung zur Partner-Anwendung aufgebaut wird.
STA	Status der Partner-Anwendung: ON: Die Partner-Anwendung ist nicht gesperrt. Es kann eine Verbindung zu ihr aufgebaut werden oder es besteht bereits eine Verbindung. OFF: Die Partner-Anwendung ist gesperrt. Es kann keine Verbindung aufgebaut werden.
CONNECT	Hier werden mehrere Informationen geliefert. 1. Spalte: Die Partner-Anwendung ist z.Zt. mit der Anwendung verbunden (Y) oder nicht verbunden (N) oder openUTM versucht gerade eine Verbindung aufzubauen (W; waiting for connection). 2. Spalte: openUTM baut die Verbindung zur Partner-Anwendung beim Start der Anwendung automatisch auf (A) oder führt beim Start der Anwendung keinen automatischen Verbindungsaufbau durch (keine Ausgabe).
CTIME	Dauer der bestehenden Verbindung in Minuten.

LETTERS	Anzahl der Nachrichten, die seit dem Start der Anwendung über die Verbindung ausgetauscht wurden, d.h. von der lokalen Anwendung gesendet oder empfangen wurden. Der Zähler wird bei jedem Start der Anwendung auf 0 zurückgesetzt.
CONB	gibt an, wie oft die Verbindung seit dem Start der Anwendung ausgefallen ist. Der CONB-Zähler wird bei jedem Start der Anwendung auf 0 zurückgesetzt.

type=KSET

Die im Folgenden dargestellte Ausgabe erfolgt nur, wenn bei der KDCDEF-Generierung in der MAX-Anweisung für den Operanden KEYVALUE ein Wert <= 255 angegeben wurde, d.h. wenn in der Anwendung Keycodes mit einer Nummer > 255 nicht erlaubt sind.

```
KSET:kset
```

	0	1	2	3	4	5	6	.	.	.	18	19
0		x			x							
20				x		x	x					
40			x								x	
60												
80	x											
100												
120 . . .												
240												

Erläuterungen zur Ausgabe

KSET Name des Keysets

In der ersten Zeile der Ausgabe werden alle Keycodes des Keysets angezeigt, die zwischen 1 und 19 liegen, in der zweiten Zeile alle Keycodes mit Nummern zwischen 20 und 39 usw.

In der letzten Zeile werden die Keycodes mit Nummern zwischen 240 und 259 angezeigt.

Die hier dargestellte Ausgabe bedeutet, dass das Keyset *kset* die Keycodes 1, 4, 23, 25, 26, 42, 58 und 80 enthält.

type=LOAD-MODULE (BS2000-Systeme)

```
LOAD-MODULE         lmodname
VERSION (GENERATED) generated element version
VERSION (PREVIOUS)  previous element version
VERSION (CURRENT)   current element version
LIBRARY             name of program library
```

```
LOAD MODE           STATIC| STARTUP| ONCALL| POOL| POOL/STARTUP| POOL/ONCALL
CHANGEABLE          YES| NO
AUTOLINK            YES| NO
PROGRAM LIST
program1            program2
program3            program4
```

Erläuterungen zur Ausgabe

LOAD-MODULE

Name des Lademoduls, bis zu 32 Zeichen lang

VERSION (GENERATED)

generierte Version des Lademoduls

VERSION (PREVIOUS)

Vorgängerversion des Lademoduls

VERSION (CURRENT)

aktuelle geladene Version des Lademoduls

LIBRARY Name der Programmbibliothek, aus der das Lademodul geladen wurde, bis zu 54 Zeichen lang

LOAD MODE Lademodus des Lademoduls, dabei bedeutet:

STATIC Das Lademodul ist statisch in das Anwendungsprogramm eingebunden.

STARTUP Das Lademodul wird beim Start der Anwendung als eigenständige Einheit nachgeladen.

ONCALL Das Lademodul wird als eigenständige Einheit nachgeladen, wenn ein Teilprogramm oder VORGANG-Exit, das/der dem Lademodul zugeordnet ist, erstmalig aufgerufen wird.

POOL Das Lademodul wird beim Start der Anwendung in den Common Memory Pool geladen. Das Lademodul enthält keine Private Slice.

POOL/STARTUP

Die Public Slice des Lademoduls wird beim Start der Anwendung in den Common Memory Pool geladen. Die zu dem Lademodul gehörende Private Slice wird anschließend in den prozesslokalen Speicher geladen.

POOL/ONCALL

Die Public Slice des Lademoduls wird beim Start der Anwendung in den Common Memory Pool geladen. Die zu dem Lademodul gehörende Private Slice wird in den prozesslokalen Speicher geladen, wenn das erste Teilprogramm aufgerufen wird, das diesem Lademodul zugeordnet ist.

CHANGEABLE

Anzeige, ob das Lademodul im Betrieb ausgetauscht werden kann oder nicht

AUTOLINK gibt an, ob das Lademodul mit der Autolink-Funktion des BLS geladen wird

PROGRAM LIST

Liste der Namen aller Teilprogramme und Datenbereiche (AREAs), die dem Lademodul zugeordnet sind. Die Liste enthält auch die Namen aller gelöschten Objekte.

type=LPAP

```
LPAP    KSET          STATUS     OUT-Q   IDLETIME    MASTER        BUNDLE
lpap    kset         ON|OFF   Q number    seconds    master         M|Y|N
```

Erläuterungen zur Ausgabe

LPAP logischer Name der Partner-Anwendung innerhalb der lokalen Anwendung (Name des LPAP-Partners)

KSET Keyset, das der Partner-Anwendung zugeordnet ist. Das Keyset legt die Zugriffsrechte der Partner-Anwendung innerhalb der lokalen Anwendung fest.

STA Status der Partner-Anwendung:

1. Spalte:
ON
Die Partner-Anwendung ist nicht gesperrt. Es kann eine Verbindung aufgebaut werden oder es besteht bereits eine Verbindung.
OFF
Die Partner-Anwendung ist gesperrt. Es kann keine Verbindung aufgebaut werden.

2. Spalte:
Q (QUIET)
Für die Partner-Anwendung werden keine Dialog-Aufträge mehr angenommen.

OUT-Q Anzahl der Nachrichten in der Message Queue, die noch an die Partner-Anwendung gesendet werden müssen.

IDLETIME Zeit bis zum Abbau einer nicht genutzten Verbindung (Session) zwischen der Partner-Anwendung und der lokalen Anwendung.

MASTER Falls der LPAP-Partner zu einem LU6.1-LPAP-Bündel gehört, wird der Name des Master-LU6.1-LPAPs des Bündels angezeigt.

BUNDLE Gibt an, ob der LPAP-Partner zu einem LU6.1-LPAP-Bündel gehört.

M
Der LPAP-Partner ist das Master-LU6.1-LPAP eines LPAP-Bündels.

Y
Der LPAP-Partner ist ein Slave-LU6.1-LPAP eines LU6.1-LPAP-Bündels.

N
Der LPAP-Partner gehört zu keinem LU6.1-LPAP-Bündel.

type=LSES

Die Ausgabe ist abhängig davon, ob einem LSES-Objekt ein kurzer oder ein langer Rechnername zugeordnet ist. Bei einem langen Rechnernamen wird die Information zu einem LSES-Objekt in zwei Bildschirmzeilen ausgegeben.

```
LSES     PRONAM   CON      BCAMAPPL RSES     LPAP    AG/USER
lses     proname  con      applname rses     lpap    user
LSES     PRONAM   CON      BCAMAPPL RSES     LPAP    AG/USER
lses     long.processor.name
                  con      applname rses     lpap    user
```

Erläuterungen zur Ausgabe

LSES — Name der LU6.1-Session innerhalb der lokalen Anwendung

RSES — Name der Session in der Partner-Anwendung

LPAP — logischer Name der Partner-Anwendung, für die die Session generiert ist

CON, PRONAM, BCAMAPPL

identifizieren die logische Verbindung eindeutig, die für die Session aufgebaut ist.

con ist der mit KC_CREATE_OBJECT Objekttyp KC_CON erzeugte oder mit der KDCDEF-Steueranweisung CON generierte Name für die logische Verbindungen zur Partner-Anwendung *lpap*.

proname ist der Name des Rechners, auf dem die Partner-Anwendung *lpap* abläuft.

applname ist der Name der lokalen UTM-Anwendung (BCAMAPPL-Name), über den die Verbindung zur Partner-Anwendung aufgebaut wird.

AG/USER — Name des Auftraggebers, für den die Session belegt ist. *user* gibt an, wer den Auftraggeber-Vorgang gestartet hat.

Läuft der Auftraggeber-Vorgang in der lokalen Anwendung ab, dann wird für *user* die Benutzerkennung oder der LTERM-Partner angegeben, die/der den Vorgang gestartet hat.

Läuft der Auftraggeber-Vorgang in der Partner-Anwendung ab (die lokale Anwendung bearbeitet den Auftrag), oder werden auf der Session asynchrone Nachrichten übertragen, dann wird für *user* der lokale Session-Name (LSES-Name) ausgegeben, d.h. die Ausgaben für LSES und AG/USER sind identisch.

type=LTAC

```
LTAC LOCK    STATUS    RTAC   CODE    LPAP  ACCESSWAIT   REPLYWAIT USED   D
ltac number  ON|OFF    rtac   I|P|T   lpap  seconds      seconds   number D
```

Erläuterungen zur Ausgabe

LTAC — Lokaler TAC-Name für das Service-Programm in der Partner-Anwendung

LOCK Lockcode, der dem fernen Vorgang in der lokalen Anwendung zugeordnet ist (Zugriffsschutz); eine Zahl zwischen 1 und 4000.

STATUS Der Transaktionscode LTAC ist gesperrt (OFF) oder nicht gesperrt (ON).

RTAC Name des Transaktionscodes/Service-Programms in der Partner-Anwendung

CODE gibt an, welcher Codetyp von openUTM intern für den RTAC-Namen verwendet wird.

- I Codetyp Integer
- P Codetyp PRINTABLE-STRING
- T Codetyp T61-String

LPAP logischer Name der Partner-Anwendung innerhalb der lokalen Anwendung (Name des LPAP-Partners)

ACCESSWAIT

Zeit, die openUTM beim Start des fernen Service-Programms auf die Belegung einer Session wartet (eventuell einschließlich Verbindungsaufbau); Angabe in Sekunden.
Ist der LTAC ein Asynchron-TAC, dann bedeutet eine Wartezeit != 0, dass der Auftrag immer in die Message Queue für die Partner-Anwendung eingetragen wird.
Die Zeit wird bei der KDCDEF-Generierung festgelegt und kann per Administration (z.B. mit dem TAC KDCAPPL) angepasst werden

REPLYWAIT Zeit, die openUTM maximal auf eine Antwort von dem fernen Service wartet. Die Zeit wird bei der KDCDEF-Generierung festgelegt und kann per Administration (z.B. mit dem TAC KDCAPPL) angepasst werden

USED Anzahl der seit Start der Anwendung erteilten Aufträge an diesen LTAC. Der Zähler wird bei jedem Start der Anwendung auf 0 zurückgesetzt.

D Gibt an, ob der LTAC per dynamischer Administration gelöscht wurde (D) oder nicht (keine Angabe).

type=LTERM

```
LTERM  PTERM  USER  KSET LOCK  USAGE       STATUS   OUT-Q   INCNT    SECCNT   D
lterm  pterm  user  kset lock  D|O         ON|OFF   number  number   number   D
                               B|M|S|
                               P|G|A
```

Erläuterungen zur Ausgabe

LTERM Name des LTERM-Partners; logischer Name des zugeordneten Client/Druckers.

PTERM Name des Client oder des Druckers (PTERM-Name), dem dieser LTERM-Partner zugeordnet ist.

USER

Benutzerkennung, die derzeit über diesen LTERM-Partner mit der Anwendung verbunden ist. Besteht derzeit keine Verbindung, dann ist *user* mit Leerzeichen belegt.

- Bei aufgebauter Verbindung:
  Ist an einem Terminal noch kein Benutzer angemeldet, ist *user* ebenfalls mit Leerzeichen belegt.
- In Anwendungen mit MULTI-SIGNON=YES:
  Ist an einem LTERM-Partner eines Clients vom Typ UPIC / APPLI / SOCKET eine reale Benutzerkennung mit RESTART=YES angemeldet, so enthält *user* diese Benutzerkennung, ansonsten den Verbindungs-Nutzer (*user*).
- In Anwendungen mit MULTI-SIGNON=NO:
  Ist an einem LTERM-Partner eines Clients vom Typ UPIC / APPLI / SOCKET eine reale Benutzerkennung angemeldet, so enthält *user* diese Benutzerkennung, ansonsten den Verbindungs-Nutzer (*user*).

KSET Keyset, das diesem LTERM-Partner zugeordnet ist (Zugriffsrechte).

LOCK Lockcode, der dem LTERM-Partner zugeordnet ist (Zugriffsschutz).

USAGE Art des LTERM-Partners

1. Wert:
D: dem LTERM-Partner ist ein Client zugeordnet oder
O: dem LTERM-Partner ist ein Drucker zugeordnet

2. Wert:
B: dem LTERM-Partner ist ein Druckerbündel zugeordnet
M: das LTERM ist ein Master-LTERM eines LTERM-Bündels
S: das LTERM ist ein Slave-LTERM eines LTERM-Bündels

3. Wert:
P: der LTERM-Partner gehört zu einem LTERM-Pool
G: das LTERM ist ein Primary-LTERM einer LTERM-Gruppe
A: das LTERM ist ein Alias-LTERM einer LTERM-Gruppe

STATUS Der LTERM-Partner ist gesperrt (OFF) oder nicht gesperrt (ON).

OUT-Q Anzahl der Nachrichten, die für den LTERM-Partner noch ausgegeben werden müssen.

INCNT Anzahl der Nachrichten, die über diesen LTERM-Partner eingegeben wurden; ist über den LTERM-Partner ein Drucker angeschlossen, dann wird hier die Anzahl der Abdruckquittungen eingetragen. Der INCNT-Zähler wird bei jedem Start der Anwendung auf 0 zurückgesetzt.

SECCNT Anzahl der Sicherheitsverletzungen an diesem LTERM-Partner seit dem Start der Anwendung (z.B. nicht erlaubter Transaktionscode eingegeben). Der SECCNT-Zähler wird bei jedem Start der Anwendung auf 0 zurückgesetzt.

D Gibt an, ob der LTERM-Partner per dynamischer Administration gelöscht wurde (D) oder nicht (keine Angabe).

type=MUX (BS2000-Systeme)

```
MUX   PRONAM   BCAMAPPL  STATUS   CONNECT  MAXSES  ACTCON  MAXCON
mux1  proname  applname  ON       Y   A    number  number  number
                         OFF      N
                                  W
```

Erläuterungen zur Ausgabe

MUX	Name des Multiplexanschlusses
PRONAM	Name des Prozessors, auf dem der Nachrichtenverteiler abläuft
BCAMAPPL	Anwendungsname der lokalen Anwendung (BCAMAPPL-Name), über den die Verbindung zu dem Multiplexanschluss aufgebaut wird.
STATUS	Der Multiplexanschluss ist gesperrt (OFF) oder nicht gesperrt (ON).
CONNECT	Hier werden mehrere Informationen geliefert. 1. Wert: Der Multiplexanschluss ist mit der Anwendung verbunden (Y) oder nicht (N) oder openUTM versucht gerade eine Verbindung zu dem Multiplexanschluss aufzubauen (W; waiting for connection) 2. Wert: Beim Start der Anwendung versucht openUTM automatisch, die Verbindung zum Multiplexanschluss aufzubauen (A) oder nicht (keine Angabe)
MAXSES	Anzahl der Terminals, die maximal gleichzeitig über diesen Multiplexanschluss an die Anwendung angeschlossen werden können.
ACTCON	Anzahl der Terminals, die derzeit über diesen Multiplexanschluss mit der Anwendung verbunden sind.
MAXCON	Maximale Anzahl der Terminals, die gleichzeitig über diesen MUX-Anschluss mit der Anwendung verbunden waren. Der MAXCON-Zähler wird bei jedem Start der Anwendung auf 0 zurückgesetzt.

MUX,OPTION=MONITORING (BS2000-Systeme)

```
MUX   PRONAM   BCAMAPPL  LETTERS  INCNT   WAIT    SHORT   RTRYO   RTRYI
mux1  proname  applname  number   number  number  number  number  number
```

Erläuterungen zur Ausgabe

MUX	Name des Multiplexanschlusses
PRONAM	Name des Prozessors, auf dem der Nachrichtenverteiler abläuft

BCAMAPPL	Anwendungsname der lokalen Anwendung (BCAMAPPL-Name), über den die Verbindung zu dem Multiplexanschluss aufgebaut wird.
LETTERS	Anzahl der Ein- und Ausgabe-Nachrichten für diesen Multiplexanschluss seit dem Start der Anwendung. Der Zähler wird bei jedem Start der Anwendung auf 0 zurückgesetzt.
INCNT	Anzahl der Eingabe-Nachrichten, die über diesen Multiplexanschluss empfangen wurden. Der INCNT-Zähler wird bei jedem Start der Anwendung auf 0 zurückgesetzt.
WAIT	Anzahl der seit Anwendungsstart von BCAM an den Multiplexanschluss übergebenen Aufforderungen, eine Nachricht nochmals zu senden, die zuvor wegen eines BCAM-Engpasses (WAIT FOR GO) von BCAM nicht angenommen werden konnte.
SHORT	Anzahl der BCAM-Engpässe (shortages) für diese Multiplexverbindung seit dem Start der Anwendung.
RTRYO	Anzahl der Versuche seit Anwendungsstart, eine Ausgabe-Nachricht nochmals zu senden (retry out).
RTRYI	Anzahl der Versuche seit Anwendungsstart, eine Eingabe-Nachricht nochmals zu lesen (retry in).

type=OSI-ASSOCIATIONS

```
ASSOC-ID  OSI-LPAP  OSI-CON  CONTWIN  CON-STATE  CONTIME   REQ-CALLSIND-CALLS
assoc-id  osi-lpap  osi-con  Y|N      CONNECTED| minutes   number    number
                                      WAIT-GO|
                                      STOP
```

Erläuterungen zur Ausgabe

ASSOC-ID	Id der Association, die der Association beim Aufbau zugeordnet wurde. Sie ist nur solange eindeutig, wie die Association aufgebaut ist. Wird diese Association abgebaut, dann wird die Id freigegeben und kann einer anderen, danach aufgebauten Association zugeordnet werden.
OSI-LPAP	Logischer Name der Partner-Anwendung (Name des OSI-LPAP-Partners), für die die Association generiert ist.
OSI-CON	Der mit der KDCDEF-Steueranweisung OSI-CON generierte Name für die logische Verbindung zur Partner-Anwendung *osi-lpap*. Wenn keine Verbindung aufgebaut ist, werden hier Leerzeichen ausgegeben.
CONTWIN	gibt an, ob die lokale Anwendung für diese Association der Contention Winner oder der Contention Loser ist.
CON-STATE	gibt den Status der Association an.
CONNECTED	Die Association ist aufgebaut.
WAIT-GO	Die Association befindet sich in der Aufbauphase. Sie wartet auf ein „GO“ von OSS.

STOP	Die Association befindet sich in der Aufbauphase. Der OSS-Aufruf *a_assrs* ist auf „STOP“ gelaufen.
CONTIME	gibt die Dauer der bestehenden Verbindung in Minuten an.
REQ-CALLS	Anzahl der Request-/Response-Presentation-Aufrufe an OSS seit dem Aufbau der Association.
IND-CALLS	Anzahl der Indication-/Confirmation-Presentation-Aufrufe an OSS seit dem Aufbau der Association.

type=OSI-CON

Die Ausgabe ist abhängig davon, ob einem OSI-CON-Objekt ein kurzer oder ein langer Rechnername zugeordnet ist. Bei einem langen Rechnernamen wird die Information zu einem OSI-CON-Objekt in zwei Bildschirmzeilen ausgegeben.

```
OSI-CON   N-SEL    T-SEL      ACC-PNT      OSI-LPAP  ACTIVE
osi-con   proname  applname   access-point osi-lpap  Y|N
OSI-CON   N-SEL    T-SEL      ACC-PNT      OSI-LPAP  ACTIVE
osi-con   long.processor.name
                   applname   access-point osi-lpap  Y|N
```

Erläuterungen zur Ausgabe

OSI-CON	Name der logischen Verbindung zur Partner-Anwendung *osi-lpap* .
OSI-LPAP	Logischer Name der Partner-Anwendung innerhalb der lokalen Anwendung (Name des OSI-LPAP-Partners), für die die Verbindung generiert ist.
T-SEL	Anwendungsname der Partner-Anwendung im lokalen System (Transport-Selektor).
N-SEL	Logischer Name des Rechners, auf dem die Partner-Anwendung abläuft (Network-Selektor).
ACC-PNT	Name des lokalen Zugriffspunktes (Access Point), über den die Verbindung *osi-con* aufgebaut wird.
ACTIVE	Y Die Verbindung *osi-con* kann genutzt werden, d.h. Nachrichten für den angegebenen OSI-LPAP-Partner werden über diese Verbindung gesendet und von ihm empfangen. N Die Verbindung *osi-con* kann nicht genutzt werden, sie ist als Ersatzverbindung reserviert.

type=OSI-LPAP

```
OSI-LPAP KSET  STA    Q  OUT-Q   IDLET    OSI-CON  ASSOC   CONN   AUTOC  BU
osi-lpap kset  ON|OFF Q  number  seconds  osi-con  number  number number M|S
```

Erläuterungen zur Ausgabe

OSI-LPAP	Logischer Name der Partner-Anwendung innerhalb der lokalen Anwendung (Name des OSI-LPAP-Partners).
KSET	Keyset, das dem OSI-LPAP-Partner zugeordnet ist. Das Keyset legt die Zugriffsrechte der Partner-Anwendung innerhalb der lokalen Anwendung fest.
STATUS	Status der Partner-Anwendung: ON Der OSI-LPAP-Partner ist nicht gesperrt. Es kann eine Verbindung zur Partner-Anwendung aufgebaut werden oder es besteht bereits eine Verbindung. OFF Der OSI-LPAP-Partner ist gesperrt. Es kann keine Verbindung zur Partner-Anwendung aufgebaut werden.
Q (Quiet):	Für den OSI-LPAP-Partner werden keine Dialog-Aufträge mehr angenommen.
OUT-Q	Anzahl der Nachrichten in der Message Queue, die noch an die Partner-Anwendung gesendet werden müssen.
IDLETIME	Zeit zur Überwachung des Leerlauf-Zustands auf Associations zwischen der Partner-Anwendung und der lokalen Anwendung.
OSI-CON	Der mit der KDCDEF-Steueranweisung OSI-CON generierte Name für die logische Verbindung zur Partner-Anwendung.
ASSOC	Maximale Anzahl paralleler Verbindungen (Associations), die gleichzeitig zu dem OSI-LPAP-Partner existieren können. Die Zahl wird bei der KDCDEF-Generierung in der OSI-LPAP-Anweisung festgelegt.
CONNECT	Anzahl der Verbindungen zu dem OSI-LPAP-Partner, die aktuell bestehen bzw. sich im Aufbau befinden.
AUTOCON	Anzahl der Verbindungen zu dem OSI-LPAP-Partner, die openUTM beim Start der Anwendung automatisch aufbauen soll.
BU(Bundle)	Gibt an, ob der OSI-LPAP-Partner zu einem OSI-LPAP-Bündel gehört. M Der OSI-LPAP-Partner ist das Master-LPAP des OSI-LPAP-Bündels. S Der OSI-LPAP-Partner ist ein Slave-LPAP des OSI-LPAP-Bündels.

type=PAGEPOOL

```
PAGEPOOL INFORMATION
percent % PAGES FOR GSSB                percent % PAGES FOR LSSB
percent % PAGES FOR TLS                 percent % PAGES FOR ULS
percent % PAGES FOR DIALOG SERVICES     percent % PAGES FOR TAC-CLASSES
percent % PAGES FOR FPUT-MANAGEMENT     percent % PAGES FOR ASYN MESSAGES
percent % PAGES FOR MSGTAC MESSAGES     percent % PAGES FOR LPUT
percent % PAGES FOR PHYS. MESSAGES      percent % PAGES FOR RESET MESSAGES
```

```
percent % PAGES FOR OSI TP LOG RECORDS   percent % OTHER PAGES
percent % FREE PAGES
```

Erläuterungen zur Ausgabe

PAGES FOR GSSB

Anzahl der Seiten in Prozent, die für GSSBs belegt sind.

PAGES FOR LSSB

Anzahl der Seiten in Prozent, die für LSSBs belegt sind.

PAGES FOR TLS

Anzahl der Seiten in Prozent, die für TLS-Bereiche belegt sind.

PAGES FOR ULS

Anzahl der Seiten in Prozent, die für ULS-Bereiche belegt sind.

PAGES FOR DIALOG SERVICES

Anzahl der Seiten in Prozent, die für Vorgangskontexte von Benutzern belegt sind.

PAGES FOR TAC-CLASSES

Anzahl der Seiten in Prozent, die für Dialogeingabenachrichten belegt sind, die temporär in TAC-Klassen Queues zwischengespeichert sind.

PAGES FOR FPUT-MANAGEMENT

Anzahl der Seiten in Prozent, die für die Verwaltung von Asynchronnachrichten belegt sind.

PAGES FOR ASYN MESSAGES

Anzahl der Seiten in Prozent, die für Asynchronnachrichten belegt sind.

PAGES FOR MSGTAC MESSAGES

Anzahl der Seiten in Prozent, die für MSGTAC-Nachrichten belegt sind.

PAGES FOR LPUT

Anzahl der Seiten in Prozent, die für temporär zwischengespeicherte LPUT-Sätze belegt sind.

PAGES FOR PHYS. MESSAGES

Anzahl der Seiten in Prozent, die für Ausgabenachrichten belegt sind, die temporär zwischengespeichert werden müssen, weil sie aufgrund ihrer Länge nur abschnittsweise an das Transportsystem übergeben werden können.

PAGES FOR RESET MESSAGES

Anzahl der Seiten in Prozent, die für Rücksetznachrichten belegt sind.

PAGES FOR OSI TP LOG RECORDS

Anzahl der Seiten in Prozent, die für OSI TP Log-Records belegt sind.

OTHER PAGES

Anzahl anderer belegter Seiten in Prozent.

FREE PAGES

Anzahl der freien Seiten in Prozent.

> Bei UTM-Cluster-Anwendungen werden GSSB- und ULS-Bereiche im globalen Pagepool der UTM-Cluster-Anwendung abgelegt. Da KDCINF PAGEPOOL nur die Belegung des lokalen Pagepools ausgibt, sind in UTM-Cluster-Anwendungen die Werte für GSSB und ULS immer Null.

type=POOL

Die Ausgabe ist abhängig davon, ob einem LTERM-Pool-Objekt ein kurzer oder ein langer Rechnername zugeordnet ist. Bei einem langen Rechnernamen wird die Information zu einem LTERM-Pool-Objekt in zwei Bildschirmzeilen ausgegeben.

```
POOL     PRONAM   BCAMAPPL PTYPE STATIONS STA=ON ACTCON MAXCON KSET     LOCK
ltprefix proname  applname ptype number   number number number kset     lock
POOL     PRONAM   BCAMAPPL PTYPE STATIONS STA=ON ACTCON MAXCON KSET     LOCK
ltprefix long.processor.name
                  applname ptype number   number number number kset     lock
```

Erläuterungen zur Ausgabe

POOL	LTERM-Präfix des LTERM-Pools. Die Namen der LTERM-Partner, die dem Pool zugeordnet werden, setzen sich zusammen aus *ltprefix* und einer Laufnummer zwischen 1 und der Maximalzahl der Clients, die sich gleichzeitig an den LTERM-Pool anschließen dürfen.
PRONAM	Nur Clients, die sich am Rechner *proname* befinden, können über den LTERM-Pool Verbindungen zur Anwendung aufbauen. Bei Anwendungen auf Unix-, Linux- und Windows-Systemen werden für *proname* Leerzeichen angegeben, wenn der LTERM-Pool für lokal angeschlossene Clients definiert ist.
BCAMAPPL	Name der lokalen Anwendung (BCAMAPPL-Name), über den die Verbindungen zu diesem LTERM-Pool aufgebaut werden (siehe KDCDEF-Anweisung TPOOL, Operand BCAMAPPL).
PTYPE	Physikalischer Typ der Clients, die sich über diesen LTERM-Pool an die Anwendung anschließen dürfen.
STATIONS	Maximale Anzahl der Clients, die sich gleichzeitig über diesen LTERM-Pool an die Anwendung anschließen können.
STA=ON	Anzahl der Clients, die derzeit für den LTERM-Pool zugelassen sind.
ACTCON	Anzahl der Clients, die derzeit über diesen Pool mit der Anwendung verbunden sind.
MAXCON	Maximale Anzahl der Clients, die im aktuellen Anwendungslauf gleichzeitig über diesen LTERM-Pool mit der Anwendung verbunden gewesen sind. Der Zähler wird bei jedem Start der Anwendung auf 0 zurückgesetzt.
KSET	Keyset, das dem LTERM-Pool und damit allen Clients, die über diesen LTERM-Pool mit der Anwendung verbunden sind, zugeordnet ist (Zugriffsrechte).
LOCK	Lockcode, der dem LTERM-Pool zugeordnet ist (Zugriffsschutz).

type=PROG

Für KDCINF PROG,L=KDCALL,CONT=*programname*

Ausgabe für UTM-Anwendungen auf BS2000-Systemen

PROGRAM	LOAD-MODULE	L-MODE	CHN	D
program1 program2	load module1 load module2	load mode load mode	YES | NO YES | NO	D

Ausgabe für UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen

PROGRAM	SHARED-OBJECT	L-MODE	CHN	D
program1 program2	shared object1 shared object2	load mode load mode	YES | NO YES | NO	D

Erläuterungen zur Ausgabe

PROGRAM Name des Teilprogramms, wie bei der Generierung in der PROGRAM-Anweisung angegeben, bis zu 32 Zeichen lang.

LOAD-MODULE

Name des Lademoduls auf BS2000-Systemen, dem dieses Teilprogramm zugeordnet ist, bis zu 32 Zeichen lang.

SHARED-OBJECT

Name des Shared Objects/DLL auf Unix-, Linux- und Windows-Systemen, dem dieses Teilprogramm zugeordnet ist, bis zu 32 Zeichen lang.

L-MODE Lademodus des Lademoduls/Shared Objects/DLL, dem dieses Teilprogramm zugeordnet ist. Dabei bedeutet:

STATIC
Das Lademodul/Shared Object/DLL ist statisch in das Anwendungsprogramm eingebunden.

STARTUP
Das Lademodul/Shared Object/DLL wird beim Start der Anwendung als eigenständige Einheit nachgeladen.

ONCALL
Das Lademodul/Shared Object/DLL wird als eigenständige Einheit nachgeladen, wenn ein Teilprogramm oder VORGANG-Exit, das/der dem Lademodul/Shared Object/DLL zugeordnet ist, erstmalig aufgerufen wird.

Nur auf BS2000-Systemen:

POOL
Das Lademodul wird beim Start der Anwendung in den Common Memory Pool geladen. Das Lademodul enthält keine Private Slice.

POOL/STARTUP
Die Public Slice des Lademoduls wird beim Start der Anwendung in den Common Memory Pool geladen. Die zu dem Lademodul gehörende Private Slice wird anschließend in den prozesslokalen Speicher geladen.

POOL/ONCALL
Die Public Slice des Lademoduls wird beim Start der Anwendung in den Common Memory Pool geladen. Die zu dem Lademodul gehörende Private Slice wird in den prozesslokalen Speicher geladen, wenn das erste Teilprogramm aufgerufen wird, dass diesem Lademodul zugeordnet ist.

CHANGEABLE

Anzeige, ob das Lademodul/Shared Object/DLL, dem dieses Teilprogramm zugeordnet ist, ausgetauscht werden kann.

D — Gibt an, ob das Programm durch die Administration aus der Konfiguration gelöscht wurde (D) oder nicht (keine Angabe).

type=PTERM

Die Ausgabe ist abhängig davon, ob einem PTERM-Objekt ein kurzer oder ein langer Rechnername zugeordnet ist. Bei einem langen Rechnernamen wird die Information zu einem PTERM-Objekt in zwei Bildschirmzeilen ausgegeben.

```
PTERM  PRONAM  LTERM  BCAMAPPL PTYP  STA     CONNECT      CTIME   LETTERS CONB   D
pterm  proname lterm  applname ptype ON|OFF  Y|N|W A|P M  minutes number  number D
                                             T|E
pterm  long.processor.name
               lterm  applname ptype ON|OFF  Y|N|W A|P M  minutes number  number D
                                             T|E
```

Erläuterungen zur Ausgabe

PTERM — Name des Client bzw. Druckers (PTERM-Name).

PRONAM — Name des Rechners, auf/an dem sich der Client/Drucker befindet.

In UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen werden für lokale Clients und Drucker Leerzeichen ausgegeben.

LTERM — Name des LTERM-Partners (logischer Name), der diesem Client/Drucker zugeordnet ist.

BCAMAPPL — Name der lokalen UTM-Anwendung (BCAMAPPL-Name), über den die Verbindung zu dem Client /Drucker aufgebaut wird.

PTYP — Typ des Client/Druckers (Bedeutung der Ausgabe siehe Tabelle im Abschnitt "kc_pterm_str - Clients und Drucker" (BS2000-Systeme, Unix-, Linux- und Windows-Systeme)

STA — Der Client/Drucker ist gesperrt (OFF) oder nicht gesperrt (ON).

CONNECT — Hier werden mehrere Informationen geliefert.

- 1. Spalte

  Y/N/W:
  Client/Drucker ist z.Zt. mit der Anwendung verbunden (Y) bzw. nicht verbunden (N) oder openUTM versucht gerade eine Verbindung aufzubauen (W; waiting for connection)

  T/E:
  wird nur für Terminals ausgegeben, die über einen Multiplexanschluss mit einer UTM-Anwendung unter einem BS2000-System verbunden sind.

  T: (timer) Die Session ist im Zustand DISCONNECT-PENDING; der Timer für das Warten auf die Bestätigung des Verbindungsaufbaus läuft.

  E: (expired) Die Session ist im Zustand DISCONNECT-PENDING und der Timer für das Warten auf die Bestätigung ist ohne Eintreffen der Bestätigung abgelaufen.

  In beiden Fällen kann die Session mit KDCPTERM freigegeben werden.
- 2. Spalte:

  A:automatischer Verbindungsaufbau beim Start der Anwendung oder

  P: der Client ist über einen LTERM-Pool mit der Anwendung verbunden.

  Wenn keine der beiden Eigenschaften zutrifft, wird hier nichts ausgegeben.
- 3. Spalte (nur bei UTM-Anwendungen auf BS2000-Systemen):

  Der Client ist über einen Multiplexanschluss mit der Anwendung verbunden (M) oder nicht (keine Angabe).

CTIME — Dauer der bestehenden Verbindung in Minuten.

LETTERS — Anzahl der Ein- und Ausgabe-Nachrichten für den Client bzw. Ausgabe-Nachrichten an den Drucker seit dem Start der Anwendung.
Der Zähler wird bei jedem Start der Anwendung auf 0 zurückgesetzt.

CONB — Anzahl der Ausfälle von Verbindungen zwischen dem Client/Drucker und der Anwendung seit dem Start der Anwendung. Der CONB-Zähler wird bei jedem Start der Anwendung auf 0 zurückgesetzt.

In UTM-Anwendungen auf BS2000-Systemen wird der CONB-Zähler auch hochgezählt, wenn ein UPIC-Client seine Verbindung zur UTM-Anwendung zunächst abbaut und danach unter demselben PTERM-Namen eine neue Verbindung aufbaut.

D — gibt an, ob der Client/Drucker durch die Administration aus der Konfiguration gelöscht wurde (D) oder nicht (keine Angabe).

type=SHARED-OBJECT (Unix-, Linux- und Windows-Systeme)

Für KDCINF SHARED-OBJECT, L=*shared-object-name*, CONT=*programname*

```
SHARED-OBJECT       shared object name
VERSION (PREVIOUS)  old version
VERSION (CURRENT)   new version
LIBRARY             name of program directory
LOAD MODE           STATIC|STARTUP|ONCALL
CHANGEABLE          YES|NO
PROGRAM LIST
program1            program2
program3            program4
```

Erläuterungen zur Ausgabe

SHARED-OBJECT

Name des Shared Objects/DLL, bis zu 32 Zeichen lang

VERSION (PREVIOUS)

Vorgängerversion des Shared Objects/DLL

VERSION (CURRENT)

aktuelle geladene Version des Shared Objects/DLL

LIBRARY Name der Programmbibliothek, aus der das Shared Object/DLL geladen wurde, bis zu 54 Zeichen lang

LOAD MODE Lademodus des Shared Objects/DLL, dabei bedeutet:

- STATIC Das Shared Object/DLL ist statisch in das Anwendungsprogramm eingebunden.
- STARTUP Das Shared Object/DLL wird beim Start der Anwendung als eigenständige Einheit nachgeladen.
- ONCALL Das Shared Object/DLL wird als eigenständige Einheit nachgeladen, wenn ein Teilprogramm oder VORGANG-Exit, das/der dem Shared Object/DLL zugeordnet ist, erstmalig aufgerufen wird.

CHANGEABLE

Anzeige, ob das Shared Object/DLL im Betrieb ausgetauscht werden kann oder nicht.

PROGRAM LIST

Liste der Namen aller Teilprogramme und Datenbereiche (AREAs), die dem Shared Object/DLL zugeordnet sind.

type=STATISTICS

```
    name   APPLINAME                    version   VERSION OF UTM
yy-mm-dd   GEN APPLICATION DATE         hh:mm:ss  GEN APPLICATION TIME
yy-mm-dd   START APPLICATION DATE       hh:mm:ss  START APPLICATION TIME
yy-mm-dd   CURRENT DATE                 hh:mm:ss  CURRENT TIME
  number   TERMINAL INPUT MESSAGES       number   TERMINAL OUTPUT MESSAGES
  number   CURRENT TASKS                 number   CONNECTED USERS
  number   OPEN DIALOG SERVICES          number   OPEN ASYN SERVICES
 percent % CURRENT LOAD                 percent % MAXIMUM LOAD
  number   DIALOG TAS PER SECOND         number   ASYN TAS PER SECOND
  number   DIALOG STEPS PER SECOND      percent % MAXIMUM POOL SIZE
 percent % ACTUAL POOL SIZE             percent % AVERAGE POOL SIZE
 percent % CACHE HIT RATE                number   NR CACHE SEARCHES
 percent % CACHE WAITS FOR BUFFER        number   NR CACHE REQUESTS
  number   UNPROCESSED ATACS             number   UNPROCESSED PRINTS
  number   WAITING DPUTS                 number   ABNORMAL TERMINATED SERVS
  number   LOGFILE WRITES                number   UTM-DEADLOCKS
  number   PERIODIC WRITES               number   PAGES PER PERIODIC WRITE
 percent % WAITS FOR RESOURCES           number   NR RESOURCE REQUESTS
 percent % MAX WAITS FOR RESOURCES       number   NR RES REQUESTS FOR MAX
 percent % WAITS FOR SYSTEM RES          number   NR SYSTEM RES REQUESTS
 percent % MAX WAITS FOR SYSTEM RES      number   NR SYSTEM RES REQ FOR MAX
 percent % ACTUAL JR                    percent % MAXIMUM JR
  number   AVG COMPRESS PAGES SAVED
```

Erläuterungen zur Ausgabe

APPLINAME

Name der Anwendung, der bei der KDCDEF-Generierung in MAX APPLINAME festgelegt wurde.

VERSION OF UTM

Eingesetzte openUTM-Version mit Korrekturstand.

GEN APPLICATION DATE

Datum des Generierungslaufs für die Anwendung.

GEN APPLICATION TIME

Uhrzeit des Generierungslaufs für die Anwendung.

START APPLICATION DATE

Tag des letzten Kaltstarts der Anwendung (UTM-S-Anwendung),
Tag des letzten Starts der Anwendung (UTM-F-Anwendung).

START APPLICATION TIME

Uhrzeit des letzten Kaltstarts der Anwendung (UTM-S-Anwendung),Uhrzeit des letzten Starts der Anwendung (UTM-F-Anwendung).

CURRENT DATE

Aktuelles Datum.

CURRENT TIME

Aktuelle Uhrzeit.

TERMINAL INPUT MESSAGES

Anzahl aller Nachrichten, die die Anwendung seit der letzten vollen Stunde von Clients oder Partner-Anwendungen empfangen hat.

TERMINAL OUTPUT MESSAGES

Anzahl aller Nachrichten, die die Anwendung seit der letzten vollen Stunde an Clients, Drucker oder Partner-Anwendungen gesendet hat.

CURRENT TASKS

Aktuelle Anzahl der Prozesse der Anwendung.

CONNECTED USERS

Anzahl der Benutzer, die derzeit mit der Anwendung verbunden sind.

OPEN DIALOG SERVICES

Anzahl der derzeit offenen Dialog-Vorgänge.

OPEN ASYN SERVICES

Anzahl der derzeit offenen Asynchron-Vorgänge.

CURRENT LOAD

Momentane Auslastung der Anwendung während des letzten abgeschlossenen Intervalls von 100 Sekunden in Prozent.

Der Wert in diesem Feld zeigt die aktuelle Auslastung der Prozesse der Anwendung durch die Bearbeitung von Aufträgen. Ist der Wert sehr hoch, sollten für die Anwendung zusätzliche Prozesse gestartet werden.

MAXIMUM LOAD

Maximale Auslastung der UTM-Anwendung seit ihrem Start oder seit dem letzten Rücksetzen des Wertes in Prozent.

DIALOG TAS PER SECOND

Anzahl der derzeit ausgeführten Dialog-Transaktionen pro Sekunde.

ASYN TAS PER SECOND

Anzahl der derzeit ausgeführten Asynchrontransaktionen pro Sekunde.

DIALOG STEPS PER SECOND

Anzahl der derzeit ausgeführten Dialog-Schritte pro Sekunde.

MAXIMUM POOL SIZE

Maximale Belegung des Pagepools in Prozent. In UTM-S-Anwendungen wird der Wert bei Neugenerierung mit KDCDEF oder bei Änderungsgenerierung mit KDCDEF/KDCUPD auf 0 gesetzt. Bei UTM-F-Anwendungen wird der Wert mit jedem Anwendungsstart auf 0 gesetzt.

ACTUAL POOL SIZE

Aktuelle Belegung des Pagepools in Prozent.

AVERAGE POOL SIZE

Mittlere Belegung des Pagepools in Prozent. In UTM-S-Anwendungen wird der Wert bei Neugenerierung mit KDCDEF oder bei Änderungsgenerierung mit KDCDEF/KDCUPD auf 0 gesetzt. Bei UTM-F-Anwendungen wird der Wert mit jedem Anwendungsstart auf 0 gesetzt.
Damit der Wert sinnvoll ist, müssen schon viele Dialog-Schritte bearbeitet worden sein.

CACHE HIT RATE

Trefferquote bei der Suche einer Seite im Cache-Speicher.
Angabe in Prozent.
CACHE HIT RATE wird bei jedem Start der Anwendung auf 0 gesetzt.

NR CACHE SEARCHES

Anzahl der Suchvorgänge nach UTM-Seiten im Cache, die für die Berechnung des Wertes CACHE HIT RATE berücksichtigt wurden.

CACHE WAITS FOR BUFFER

Anforderungen von Puffern im Cache, die zu einer Wartezeit geführt haben.Angabe in Prozent.
CACHE WAITS FOR BUFFER wird zu jeder vollen Stunde auf 0 gesetzt.

NR CACHE REQUESTS

Anzahl von Pufferanforderungen, die für die Berechnung des Werts CACHE WAITS FOR BUFFER berücksichtigt wurden.

UNPROCESSED ATACS

Anzahl der Nachrichten für Asynchron-Vorgänge, die derzeit in openUTM gespeichert und noch nicht vollständig bearbeitet sind.

UNPROCESSED PRINTS

Anzahl der derzeit an den Druckern anstehenden Nachrichten.

WAITING DPUTS

Anzahl der derzeit wartenden zeitgesteuerten Aufträge (DPUTs).

ABNORMAL TERMINATED SERVS

Anzahl der abnormal beendeten Vorgänge. In UTM-S-Anwendungen wird der Wert bei Neugenerierung mit KDCDEF oder bei Änderungsgenerierung mit KDCDEF/KDCUPD auf 0 gesetzt. Bei UTM-F-Anwendungen wird der Wert mit jedem Anwendungsstart auf 0 gesetzt.

LOGFILE WRITES

Anzahl der Anforderungen, auf die Benutzer-Protokolldatei (USLOG) zu schreiben.
Der Zähler LOGFILE WRITES wird zu jeder vollen Stunde auf 0 gesetzt.

UTM-DEADLOCKS

Anzahl der erkannten und aufgelösten Deadlocks von UTM-Betriebsmitteln.
In UTM-S-Anwendungen wird der Wert bei Neugenerierung mit KDCDEF oder bei Änderungsgenerierung mit KDCDEF/KDCUPD auf 0 gesetzt. Bei UTM-F-Anwendungen wird der Wert mit jedem Anwendungsstart auf 0 gesetzt.

PERIODIC WRITES

Anzahl der Periodic Writes seit dem letzten Start der Anwendung (periodic write = Sicherung der gesamten sicherungsrelevanten Verwaltungsdaten der UTM-Anwendung).

PAGES PER PERIODIC WRITE

Anzahl der UTM-Seiten, die bei einem periodic write im Mittel gesichert wurden.
Der Zähler wird bei jedem Start der Anwendung auf 0 gesetzt.

WAITS FOR RESOURCES

Dieser Wert gibt die mittlere Lockkonfliktrate der Speicherbereiche GSSB, ULS und TLS im letzten 100 Sekunden Intervall in der Einheit Promille an, d.h. die Anzahl Wartesituationen bei Lockanforderungen pro Anzahl der Lockanforderungen für GSSB, ULS und TLS insgesamt im letzten 100 Sekunden Intervall multipliziert mit 1000.

Ein hoher Wert für WAITS FOR RESOURCES kann folgende Ursachen haben:

- Prozesse mit zu langen Laufzeiten oder Wartezeiten,
- Betriebsmittel sind zu lange gesperrt, z.B. häufige PEND KP- oder PGWT-Aufrufe in KDCS-Teilprogrammen.

> **i** Geht ein Lock-Halter in den Status PEND KP, so werden alle "Waiter" benachrichtigt und alle weiteren Sperren sofort abgewiesen. D.h. der Wert von WAITS FOR RESOURCES erhöht sich dadurch nicht.

NR RESOURCE REQUESTS

Anzahl der Anforderungen an Transaktions-Ressourcen im letzten 100 Sekunden Intervall, die für die Berechnung des Wertes WAITS FOR RESOURCES berücksichtigt wurden.

MAX WAITS FOR RESOURCES

Maximale Konfliktrate für Locks auf Anwenderdaten über den Anwendungslauf. Der Wert wird in Prozent angegeben.

NR RES REQUESTS FOR MAX

Anzahl der Anforderungen an Transaktions-Ressourcen in dem 100 Sekunden Intervall, in dem die maximale Konfliktrate MAX WAITS FOR RESOURCES erreicht wurde.

WAITS FOR SYSTEM RES

Mittlere Konfliktrate im letzten 100 Sekunden Intervall für die in diesem Intervall am höchsten belastete System-Ressource. Die Ausgabe kann sich in unterschiedlichen Intervallen auf unterschiedliche System-Ressourcen beziehen. Der Wert wird in Prozent angegeben.

NR SYSTEM RES REQUESTS

Anzahl der Anforderungen an System-Ressourcen im letzten 100 Sekunden Intervall, die für die Berechnung des Wertes WAITS FOR SYSTEM RES berücksichtigt wurden.

MAX WAITS FOR SYSTEM RES

Maximale Konfliktrate für Anforderungen an System-Ressourcen (Systemlocks) über den Anwendungslauf. Der Wert wird in Prozent angegeben.

NR SYSTEM RES REQ FOR MAX

Anzahl der Anforderungen an System-Ressourcen in dem 100 Sekunden Intervall, in dem die maximale Konfliktrate MAX WAITS FOR SYSTEM RES erreicht wurden.

ACTUAL JR

Nur bei verteilter Verarbeitung:
Aktuelle Anzahl der gleichzeitig adressierten Auftragnehmer-Vorgänge relativ zum Generierungswert MAXJR, Angabe in Prozent.
(MAXJR = maximal Anzahl der Auftragnehmer-Vorgänge, die gleichzeitig in der lokalen Anwendung adressiert sein dürfen; dies entspricht der Anzahl gleichzeitig aktiver APRO-Aufrufe)

MAXIMUM JR Nur bei verteilter Verarbeitung:

Maximale Anzahl der in der lokalen Anwendung gleichzeitig adressierten fernen Auftragnehmer-Vorgänge relativ zum Generierungswert MAXJR (KDCDEF-Steueranweisung UTMD). In UTM-S-Anwendungen wird der Wert bei Neugenerierung mit KDCDEF oder bei Änderungsgenerierung mit KDCDEF/KDCUPD auf 0 gesetzt. Bei UTM-F-Anwendungen wird der Wert mit jedem Anwendungsstart auf 0 gesetzt.

Die Angabe erfolgt in Prozent.

AVG COMPRESS PAGES SAVED

Durchschnittswert der pro Datenkomprimierung eingesparten UTM- Seiten. Nicht berücksichtigt in diesem Statistikwert wird das Schreiben von Datenbereichen, bei denen UTM keine Komprimierung durchführt, weil z.B. die Datenlänge kleiner als eine UTM-Seite ist.

Wenn kein Statistikwert zur Datenkomprimierung vorliegt, dann wird statt eines numerischen Wertes die Zeichenfolge "- - -" ausgegeben. Dies ist in folgenden Situationen möglich:

- Die Datenkomprimierung ist ausgeschaltet.
- Der Wert wurde zurückgesetzt, z.B. mit KC_MODIFY_OBJECT oder per WinAdmin oder WebAdmin.
- Es wurde keine Datenkomprimierung durchgeführt, weil die Anwendung "kleine" Datenbereiche verwendet, bei denen eine Komprimierung nicht sinnvoll eingesetzt werden kann.

Ist der bei AVG COMPRESS PAGES SAVED ausgegebene Wert kleiner als 0,5, dann sollte die Datenkomprimierung für diese Anwendung aus Performance-Gründen ausgeschaltet werden.

Lebensdauer der bei STATISTICS ausgegebenen Statistikdaten

Die folgenden Statistikdaten werden beim Start der Anwendung und zu jeder vollen Stunde (wenn MAX STATISTIC-MSG=FULL-HOUR generiert ist, auch bei UTM-F-Anwendungen) aktualisiert. Der Tabelle können Sie entnehmen, wann openUTM bei einer UTM-S-Anwendung die Zähler auf 0 zurücksetzt. Bei UTM-F-Anwendungen werden alle Zähler bei jedem Anwendungsstart auf 0 gesetzt.

Einen Teil der Statistikwerte können Sie auch über die Programmschnittstelle zur Administration auf 0 zurücksetzen (siehe Abschnitt "obj_type = KC_CURR_PAR").

Zeitpunkt des Zurücksetzens	**Zähler, der zurückgesetzt wird**
bei jedem Start der Anwendung	CACHE HIT RATE PAGES PER PERIODIC WRITE PERIODIC WRITES
bei Neugenerierung mit KDCDEF und Änderungsgenerierung mit KDCDEF/KDCUPD	AVERAGE POOL SIZE MAXIMUM JR MAXIMUM POOL SIZE WAITS FOR RESOURCES
beim Start der Anwendung und zu jeder vollen Stunde (wenn MAX STATISTIC-MSG= FULL-HOUR generiert ist; auch bei UTM-F-Anwendungen)	CACHE WAITS FOR BUFFER LOGFILE WRITES UTM-DEADLOCKS ABNORMAL TERMINATED CONVS TERMINAL INPUT MESSAGES TERMINAL OUTPUT MESSAGES

Folgende Statistikwerte werden zu jeder vollen Stunde und bei normaler Beendigung der Anwendung in die Systemprotokolldatei SYSLOG geschrieben (Meldung K081), sofern die Anwendung mit MAX STATISTICS-MSG=FULL-HOUR generiert ist:

CACHE HIT RATE
CACHE WAITS FOR BUFFER
CONNECTED USERS
LOGFILE WRITES
TERMINAL INPUT MESSAGES
TERMINAL OUTPUT MESSAGES
UNPROCESSED ATACS

type=SYSLOG

Die Ausgabe der Informationen ist identisch mit der Ausgabe von KDCSLOG INFO (siehe Abschnitt "KDCSLOG - SYSLOG-Datei administrieren").

type=SYSPARM

```
appliname   APPLINAME                    version   VERSION OF UTM
   ON|OFF   ACCOUNT                       ON|OFF   CALC FOR ACCOUNTING
   ON|OFF   SM2                           ON|OFF   KDCMON
   ON|OFF   TESTMODE                    percent %  MAX CACHE PAGING RATE
   number   PROGRAM FGG                   seconds   TERMWAIT
   number   USLOG FGG                     seconds   RESWAIT-TA
   number   MAX TASKS                     seconds   RESWAIT-PR
   number   CURRENT TASKS                 seconds   CONRTIME
   number   MAXASYN TASKS                 seconds   LOGACKWAIT
   number   CURRENT MAXASYN TASKS          number   CURRENT MAX TASKS IN PGWT
  seconds   PTCTIME                       seconds   CONCTIME
  seconds   PGWTTIME                       number   TASKS WAITING IN PGWT
   YES|NO   PROGRAM EXCHANGE IS RUNNING    number   MAX TASKS IN PGWT
   YES|NO   CLUSTER-APPLICATION             PS|DS   CACHE LOCATION
   ON|OFF   DATA COMPRESSION (GEN)
```

Erläuterungen zur Ausgabe

APPLINAME — Name der Anwendung, der bei der KDCDEF-Generierung in MAX APPLINAME festgelegt wurde.

VERSION OF UTM

Eingesetzte openUTM-Version einschließlich Korrekturstand und Generierungsvariante der Anwendung (UTM-S oder UTM-F).

ACCOUNT — Die Abrechnungsphase des Accounting ist eingeschaltet (ON) oder ausgeschaltet (OFF). Kann im laufenden Betrieb ein- und ausgeschaltet werden (z.B. mit KDCAPPL).

CALC FOR ACCOUNTING

Die Kalkulationsphase des Accounting ist eingeschaltet (ON) oder ausgeschaltet (OFF). Kann im laufenden Betrieb ein- und ausgeschaltet werden (z.B. mit KDCAPPL).

SM2 — Die Datenlieferung an openSM2 ist für die Anwendung eingeschaltet (ON) bzw. ausgeschaltet (OFF).
Kann im laufenden Betrieb ein- und ausgeschaltet werden (z.B. mit KDCAPPL).

KDCMON — Der Messmonitor KDCMON ist eingeschaltet (ON) oder ausgeschaltet (OFF).
Kann im laufenden Betrieb ein- und ausgeschaltet werden (z.B. mit KDCDIAG).

TESTMODE — Der Testmodus ist eingeschaltet (ON) oder ausgeschaltet (OFF).
Kann im laufenden Betrieb ein- und ausgeschaltet werden (z.B. mit KDCDIAG).

MAX CACHE PAGING RATE

Aktuell eingestellter Wert für CACHE. Die Paging Rate gibt an, wieviel Prozent der Seiten im Cache-Speicher in Engpasssituationen maximal auf KDCFILE geschrieben werden sollen.
Der Wert kann geändert werden, z.B. mit KDCAPPL CACHE.

PROGRAM FGG

Generierungsnummer des aktuell geladenen Anwendungsprogramms.

Für UTM-Anwendungen auf BS2000-Systemen wird für PROGRAM FGG immer der Wert 0 ausgegeben.

Für UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen, die nicht aus dem Dateigenerationsverzeichnis PROG gestartet wurden, wird für PROGRAM FGG der Wert 0 ausgegeben.

TERMWAIT — Zeit in Sekunden, die bei einer Mehrschritt-Transaktion (Programm ruft PEND KP auf) maximal zwischen einer Ausgabe an einen Dialog-Partner und der nachfolgenden Dialog-Antwort verstreichen darf.

USLOG FGG — Nummer der Dateigeneration der Benutzer-Protokolldatei (USLOG), in die aktuell geschrieben wird.

RESWAIT-TA — Aktuell eingestellter Wert für die Zeit in Sekunden, die maximal auf ein von einer anderen Transaktion gesperrtes Betriebsmittel (GSSB, ULS, TLS) gewartet wird.

MAX TASKS — Maximale Anzahl der in dieser Anwendung erlaubten Prozesse (siehe Datenstruktur *kc_tasks_par_str* im Abschnitt "kc_tasks_par_str - Anzahl der Prozesse", Parameter *tasks*).

RESWAIT-PR — Aktuell eingestellter Wert für die Zeit in Sekunden, die maximal auf ein von einem anderen Prozess gesperrtes Betriebsmittel gewartet wird.

CURRENT TASKS

Aktuelle Anzahl der Prozesse der Anwendung (siehe Datenstruktur *kc_tasks_par_str* im Abschnitt "kc_tasks_par_str - Anzahl der Prozesse" , Parameter *curr_tasks*).

CONRTIME — Aktuell eingestellter Wert für die Zeit in Minuten, nach der openUTM bei Verbindungsausfall (zyklisch) versuchen soll, die Verbindung wieder aufzubauen.

MAXASYN TASKS

maximale Anzahl der Prozesse der Anwendung, die gleichzeitig für die Asynchronverarbeitung verwendet werden können (siehe Datenstruktur *kc_tasks_par_str* im Abschnitt "kc_tasks_par_str - Anzahl der Prozesse", Parameter *asyntasks*).

LOGACKWAIT

Zeit in Sekunden, die UTM-Anwendungen auf BS2000-Systemen maximal auf eine Abdruck- bzw. Transportquittung warten.

Für UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen ist die Ausgabe irrelevant.

CURRENT MAXASYN TASKS

Aktuelle Maximalzahl der Prozesse, die gleichzeitig für die Asynchronverarbeitung verwendet werden können (siehe Datenstruktur *kc_tasks_par_str* im Abschnitt "kc_tasks_par_str - Anzahl der Prozesse", Parameter *curr_max_asyntasks*).

Dieser Wert wird automatisch angepasst, wenn:

- der Wert explizit durch die Administration geändert wird (z.B. durch KDCAPPL ASYNTASKS=).
- die Anzahl der Prozesse, in denen das Anwendungsprogramm abläuft (CURRENT TASKS) geändert wird (z.B. durch KDCAPPL TASKS=). Beim Herabsetzen der Anzahl CURRENT TASKS wird auch die Anzahl CURRENT MAXASYN TASKS herabgesetzt, sobald CURRENT TASKS < CURRENT MAXASYN TASKS ist.
  Bei einem späteren Heraufsetzen der Anzahl CURRENT TASKS wird auch der Wert von CURRENT MAXASYN TASKS durch openUTM automatisch wieder heraufgesetzt.

CURRENT MAX TASKS IN PGWT

Aktuelle Maximalzahl der Prozesse der Anwendung, die gleichzeitig Teilprogramme mit blockierenden Aufrufen bearbeiten dürfen (siehe Datenstruktur *kc_tasks_par_str* im Abschnitt "kc_tasks_par_str - Anzahl der Prozesse", Parameter *curr_max_tasks_in_pgwt*).

Dieser Wert wird automatisch geändert, wenn:

- der Wert explizit durch die Administration geändert wird (z.B. durch KDCAPPL TASKS-IN-PGWT=).
- die Anzahl der Prozesse der Anwendung (CURRENT TASKS) per Administration geändert wird (z.B. durch KDCAPPL TASKS=). Beim Herabsetzen der Anzahl CURRENT TASKS wird auch die Anzahl CURRENT MAX TASKS IN PGWT herabgesetzt, sobald CURRENT TASKS <= CURRENT MAX TASKS IN PGWT ist.
  Bei einem späteren Heraufsetzen der Anzahl CURRENT TASKS wird der Wert von CURRENT MAX TASKS IN PGWT durch openUTM dann automatisch wieder heraufgesetzt.

PTCTIME — Nur bei verteilter Verarbeitung:
Zeit in Sekunden, die ein lokaler Auftragnehmer-Vorgang maximal im Zustand PTC (prepare to commit, Transaktionsstatus P) auf eine Quittung vom Auftraggeber-Vorgang wartet.
Der Wert 0 bedeutet, dass beliebig lange Zeit auf eine Quittung gewartet wird.

CONCTIME — Nur bei verteilter Verarbeitung:
Zeit in Sekunden zur Überwachung des Aufbaus einer Session (LU6.1) oder Association (OSI TP). Wenn die Session bzw. Association nicht innerhalb der angegebenen Zeit aufgebaut wird, baut openUTM die Transportverbindung ab. Damit wird verhindert, dass eine Transportverbindung wegen eines misslungenen Aufbaus einer Session oder Association blockiert bleibt. CONCTIME=0 bedeutet bei LU6.1-Verbindungen, dass der Aufbau einer Session nicht überwacht wird (es wird beliebig lange gewartet), und bei OSI TP-Verbindungen, dass maximal 60 Sekunden auf den Aufbau einer Association gewartet wird.

PGWTTIME — Zeit in Sekunden, die ein blockierender Funktionsaufruf, z.B. der KDCS- Aufruf PGWT, maximal warte

TASKS WAITING IN PGWT

Aktuelle Anzahl von Prozessen, die sich aufgrund blockierender Funktionsaufrufe (z.B. KDCS-Aufruf PGWT) im Wartezustand befinden.

PROGRAM EXCHANGE IS RUNNING

gibt an, ob openUTM gerade einen Programmaustausch für die Anwendung durchführt.

MAX TASKS IN PGWT

Anzahl der Prozesse der Anwendung, die maximal gleichzeitig Teilprogramme bearbeiten dürfen, die blockierende Funktionsaufrufe (z.B. KDCS-Aufruf PGWT) durchlaufen (siehe Datenstruktur *kc_tasks_par_str* im Abschnitt "kc_tasks_par_str - Anzahl der Prozesse", Parameter *tasks_in_pgwt*).

CLUSTER-APPLICATION

Gibt an, ob es sich um eine UTM-Cluster-Anwendung oder eine stand-alone UTM-Anwendung handelt.

CACHE LOCATION

Gibt an, ob der UTM-Cache im Programmraum (PS) oder in einem oder mehreren Datenräumen (DS) liegt.

Für Unix-, Linux- und Windows-Systeme wird immer PS angezeigt.

DATA-COMPRESSION (GEN)

Gibt an, ob für die Anwendung Datenkomprimierung erlaubt ist (ON) oder nicht (OFF). Der hier ausgegebene Wert entspricht dem Generierungswert für die Anwendung (siehe openUTM-Handbuch „Anwendungen generieren“, MAX DATA-COMPRESSION=). Wird hier ON angezeigt, dann kann die Datenkomprimierung administrativ ein- oder ausgeschaltet werden, z.B. mit KDCAPPL.

type=TAC

```
TAC  LOCK    STAT TCL    IN-Q   USED   ERROR  DBCNT  TACELAP DBELAP TACCPU D
tac  number  ON   number number number number number msec    msec   mcsec  D
             OFF  type
             HLT
             KP
```

Erläuterungen zur Ausgabe

TAC	TAC-Name
LOCK	Lockcode, mit dem der Transaktionscode zugriffsgesichert ist; Zahl zwischen 1 und 4000.
STAT	Status des Transaktionscodes: Der TAC ist nicht gesperrt (ON), gesperrt (OFF), vollständig gesperrt (HLT) oder blockiert (KP). Blockiert heißt, der TAC ist gesperrt, aber es werden Aufträge für den TAC angenommen und in die Auftragswarteschlange gestellt.

TCL	TAC-Klasse und Typ (`D | A | Q`) des Transaktionscodes bzw. der TAC-Queue.
IN-Q	Anzahl der Nachrichten, die von dem mit TAC-Name gestarteten Teilprogramm noch bearbeitet werden müssen.
USED	Anzahl der Teilprogrammläufe, die mit diesem Transaktionscode insgesamt bearbeitet wurden (nur für Asynchron-TACs). In UTM-S-Anwendungen wird der Wert bei Neugenerierung mit KDCDEF oder bei Änderungsgenerierung mit KDCDEF/KDCUPD auf 0 gesetzt. Bei UTM-F-Anwendungen wird der Wert mit jedem Anwendungsstart auf 0 gesetzt.
ERROR	Anzahl der Teilprogrammläufe, die über diesen Transaktionscode gestartet und fehlerhaft beendet wurden. In UTM-S-Anwendungen wird der Wert bei Neugenerierung mit KDCDEF oder bei Änderungsgenerierung mit KDCDEF/KDCUPD auf 0 gesetzt. Bei UTM-F-Anwendungen wird der Wert mit jedem Anwendungsstart auf 0 gesetzt.
DBCNT	Mittlere Anzahl der Datenbankaufrufe aus Teilprogrammen, die mit diesem TAC-Namen gestartet wurden. Bei Datenbankkopplungen über die XA-Schnittstelle ist DBCNT immer 0.
TACELAP	Mittlere Laufzeit der Teilprogramme, wenn sie mit diesem TAC gestartet wurden (elapsed time); Angabe in Millisekunden.
DBELAP	Mittlere Zeit für die Bearbeitung der Datenbankaufrufe in den Teilprogrammläufen mit diesem TAC; Angabe in Millisekunden. Bei Datenbankkopplungen über die XA-Schnittstelle ist DBELAP immer 0.
TACCPU	Durchschnittliche CPU-Zeit in Mikrosekunden, die zur Bearbeitung dieses Transaktionscodes im Teilprogramm verbraucht wurde. Dies entspricht der von openUTM verbrauchten CPU-Zeit plus der ggf. vom Datenbanksystem verbrauchten CPU-Zeit.
D	Gibt an, ob der Transaktionscode per Administration aus der Konfiguration gelöscht wurde (D) oder nicht (keine Angabe).

Die für *type*=TAC ausgegebenen Statistikwerte USED, ERROR, DBCNT, TACELAP, DBELAP und TACCPU werden in UTM-S-Anwendungen nur bei jeder Neugenerierung mit KDCDEF und bei jeder Änderungsgenerierung mit KDCDEF/KDCUPD auf 0 zurücksetzt; in UTM-F-Anwendungen werden die Werte bei jedem Anwendungsstart auf 0 gesetzt.

type=TAC-PROG

Ausgabe für UTM-Anwendungen auf BS2000-Systemen

```
TAC       PROGRAM            LOAD-MODULE
tac1      program1           load-module1
tac2      program2           load-module2
tac3      program3           load-module3
```

Ausgabe für UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen

```
TAC       PROGRAM            SHARED-OBJECT
tac1      program1           shared-object1
tac2      program2           shared-object2
tac3      program3           shared-object3
```

Erläuterungen zur Ausgabe

TAC Name des Transaktionscodes

PROGRAM Name des Teilprogramms, dem dieser Transaktionscode zugeordnet ist

LOAD-MODULE

Auf BS2000-Systemen: Name des Lademoduls , in dem das Teilprogramm (PROGRAM) enthalten ist

SHARED-OBJECT

Auf Unix-, Linux- und Windows-Systemen: Name des Shared Objects/DLL, in dem das Teilprogramm (PROGRAM) enthalten ist

type=TACCLASS

```
TACCLASS  TASKS  WT MSGS   AVG-WAIT-TIME PGWT    PRIO* NR     WAITS
    1     number   number  msec          YES|NO  prio  number number
    :
    8     number   number  msec          YES|NO  prio  number number
    9     number   number  msec          YES|NO  prio  number number
    :
   16     number   number  msec          YES|NO  prio  number number
```

*prio = ABS | REL | EQ | NO

Erläuterungen zur Ausgabe

TASKS maximale Anzahl der Prozesse, die zur Zeit Aufträge für Transaktionscodes dieser TAC-Klasse bearbeiten dürfen.

WT MSGS Anzahl der Nachrichten, die derzeit für Transaktionscodes dieser TAC-Klasse in openUTM gespeichert und noch nicht bearbeitet sind.

AVG-WAIT-TIME

mittlere Wartezeit von Aufträgen an diese TAC-Klasse in Millisekunden. Die Wartezeit wird von der Entgegennahme des Auftrags durch openUTM bis zum Beginn der Verarbeitung gerechnet. AVG-WAIT-TIME=0 bedeutet, dass alle Aufträge sofort bearbeitet werden.

Zu Wartezeiten kommt es, wenn nicht alle Prozesse der Anwendung Aufträge für die TAC-Klasse bearbeiten dürfen, und openUTM deshalb Aufträge in der Auftragswarteschlange zwischenspeichern muss.

PGWT gibt an, ob in dieser TAC-Klasse Teilprogramme ablaufen dürfen, die blockierende Aufrufe wie z.B. den KDCS-Aufruf PGWT enthalten.

Ist die Anwendung mit Prioritätensteuerung (TAC-PRIORITIES-Anweisung) generiert, dann enthält die Spalte PGWT für alle TAC-Klassen den Wert NO.

PRIO Ist die Anwendung mit Prioritätensteuerung generiert, dann enthält die Spalte PRIO die Art der für die TAC-Klassen generierten Prioritäten (ABS, REL, EQ). Ist die Anwendung ohne TAC-PRIORITIES-Anweisung generiert, dann enthält die Spalte PRIO für alle TAC-Klassen den Wert NO.

NR Anzahl von Teilprogrammläufen für diese TAC-Klasse.

WAITS Anzahl von Wartesituationen, die für die Berechnung des Wertes AVG-WAIT-TIME berücksichtigt wurden.

type=USER

```
USER   KSET   STATUS   OSERV  NR.TACS  CPUTIME   SECCNT   LTERM   D
user1  kset1  ON| OFF  Y| N   number   msec      number   lterm1  D
```

Erläuterungen zur Ausgabe

USER Name der Benutzerkennung

KSET Keyset, das dieser Benutzerkennung zugeordnet ist (Zugriffsrechte)

STATUS Benutzerkennung gesperrt (OFF) oder nicht gesperrt (ON).

OSERV "OSERV" steht für "open Services".

Y bedeutet, dass der Benutzer z.Zt. einen Vorgang bearbeitet und dieser Vorgang mindestens einen Sicherungspunkt erreicht hat.

N bedeutet, dass der Benutzer zur Zeit keinen Vorgang bearbeitet, der nicht schon mindestens einen Sicherungspunkt erreicht hat.

NR.TACS Anzahl der unter dieser Benutzerkennung ausgeführten Teilprogrammabläufe. In UTM-S-Anwendungen wird der Zähler bei Neugenerierung mit KDCDEF oder bei Änderungsgenerierung mit KDCDEF/KDCUPD auf 0 gesetzt. Bei UTM-F-Anwendungen wird der Zähler mit jedem Anwendungsstart auf 0 gesetzt.

CPUTIME Anzahl CPU-Millisekunden, die für die Abarbeitung der Aufträge für diesen Benutzer verbraucht wurden (die CPU-Zeit der Datenbankaufrufe ist darin nicht enthalten). Der Wert wird nach dem Abmelden des Benutzers (KDCOFF) bzw. nach Verbindungsabbau wieder auf 0 gesetzt.

SECCNT Anzahl der Sicherheitsverletzungen für diese Benutzerkennung (z.B. falsches Passwort eingegeben) seit dem Start der Anwendung. Diese Zahl wird bei jedem Start der Anwendung auf 0 gesetzt.

LTERM Folgende Fälle sind zu unterscheiden:

- Anwendungen mit MULTI-SIGNON=NO (d.h. Mehrfachanmeldungen sind nicht erlaubt): LTERM- oder OSI-LPAP-Partner, über den ein Benutzer mit dieser Benutzerkennung angemeldet

ist.
Ausnahme: LTERM enthält Leerzeichen, wenn die Anmeldung zum Starten eines Asynchron-Vorgangs über OSI TP erfolgte.

- Anwendungen mit MULTI-SIGNON=YES (Mehrfachanmeldung erlaubt):
  Ist ein Benutzer mit der Benutzerkennung über ein Terminal mit der Anwendung verbunden, dann enthält LTERM den Namen des LTERM-Partners, der dem Terminal zugeordnet ist.

  Ist die Benutzerkennung mit RESTART=YES generiert, dann enthält LTERM den Namen des LTERM- oder OSI-LPAP-Partners, über den ein Client mit dieser Benutzerkennung angemeldet ist.

  Ausnahmen: Die Anmeldung erfolgte über OSI TP und es wurde die Functional Unit „Commit“ ausgewählt oder die Anmeldung erfolgte über OSI TP zum Starten eines Asynchron-Vorgangs. Dann enthält LTERM Leerzeichen.

  In allen anderen Fällen enthält LTERM Leerzeichen.

D Gibt an, ob die Benutzerkennung per Administration aus der Konfiguration gelöscht wurde (D) oder nicht (keine Angabe).

Das Passwort der Benutzerkennung wird nicht ausgegeben.

12.6 KDCLOG - Benutzer-Protokolldatei umschalten

Die Benutzer-Protokolldatei USLOG wird als Dateigenerationsverzeichnis USLOG geführt. Mit KDCLOG können Sie im laufenden Betrieb der Anwendung die aktuelle Benutzer-Protokolldatei (Dateigeneration) schließen und gleichzeitig eine neue Benutzer-Protokolldatei eröffnen. Das ist die Dateigeneration mit der nächstfolgenden Generationsnummer. Die geschlossene Protokolldatei ist dann beliebig verwendbar. KDCLOG wirkt auf beide Dateien, wenn die Benutzer-Protokolldatei doppelt geführt wird. Zur Benutzer-Protokolldatei USLOG siehe auch das openUTM-Handbuch „Einsatz von UTM-Anwendungen“.

Wirkung in UTM-Cluster-Anwendungen

KDCLOG wirkt in UTM-Cluster-Anwendungen global, d.h. für alle laufenden Knoten-Anwendungen.

Wirkungsdauer der Änderungen

Es wird solange in die neue(n) USLOG-Dateigeneration(en) geschrieben, bis mit KDCLOG auf die nächste Dateigeneration umgeschaltet wird.

Nach Anwendungsende kann auch mit Betriebssystem-Kommandos auf die nächste Dateigeneration umgeschaltet werden (siehe openUTM-Handbuch „Einsatz von UTM-Anwendungen“).

```
KDCLOG
```

Für die Administration über Message Queuing müssen Sie KDCLOGA angeben.

KDCLOG hat keine Operanden.

Ausgabe von KDCLOG

Am Administrator-Terminal wird die Meldung

```
"COMMAND ACCEPTED"
```

angezeigt.

12.7 KDCLPAP - Verbindungen zu (OSI-)LPAP-Partnern administrieren

Mit KDCLPAP können Sie folgende Aktionen durchführen:

- Aufbau von Verbindungen veranlassen
- Verbindungen abbauen
- Verbindungen sperren bzw. gesperrte Verbindungen wieder freigeben
- Partner-Anwendungen festlegen, zu denen openUTM bei jedem Start der Anwendung automatisch Verbindungen aufbauen soll
- Anzahl paralleler Verbindungen zu OSI TP-Partner-Anwendungen festlegen
- Ersatzverbindungen zu OSI TP-Partner-Anwendungen aktivieren. Die Ersatzverbindungen müssen mit KDCDEF generiert worden sein
- Zeit zur Überwachung der Leerlauf-Zustände von Sessions und Associations ändern

Die Verbindungen werden angegeben über den Namen des LPAP- bzw. OSI-LPAP-Partners, dem sie zugeordnet sind. Ersatzverbindungen werden durch den in der KDCDEF-Steueranweisung OSI-CON definierten Namen der Ersatzverbindung identifiziert.

Besonderheiten beim Verbindungsaufbau und -abbau

Mit KDCLPAP ...,ACT=CON oder (CON,*number*) wird der Verbindungsaufbau nur veranlasst. Eine erfolgreiche Kommandoausführung bedeutet also nicht, dass die Verbindungen aufgebaut sind bzw. wirklich aufgebaut werden können (Fehler im Transportsystem). Ob openUTM eine Verbindung tatsächlich aufbauen konnte, sollten Sie deshalb mit KDCINF kontrollieren. Zum Beispiel durch:

`KDCINF OSI-LPAP, LIST=(osi-lpapname_1,...osi-lpapname_10)` für OSI-LPAP-Partner

`KDCINF CON, LIST=KDCCON` für LPAP-Partner

Wollen Sie eine Verbindung zu einem gesperrten LPAP- bzw. OSI-LPAP-Partner (STATUS=OFF) aufbauen, dann müssen Sie KDCLPAP zweimal aufrufen:

1. Ein KDCLPAP-Aufruf zum Entsperren des (OSI-)LPAP-Partners, z.B.:
   `KDCLPAP [OSI-]LPAP=lpapname,STATUS=ON`
2. Ein KDCLPAP-Aufruf, der veranlasst, dass die Verbindung aufgebaut wird, z.B.:
   `KDCLPAP [OSI-]LPAP=lpapname,ACTION=CON`

KDCLPAP mit ACTION=CON **und** STATUS=ON wird nicht bearbeitet.

Wollen Sie die Anzahl paralleler Verbindungen zu einem OSI-LPAP-Partner herabsetzen, dann rufen Sie KDCLPAP mit ACTION=(CON,*number*) auf, wobei Sie für *number* die Anzahl der Verbindungen angeben, die erhalten bleiben sollen.

Wirkung in UTM-Cluster-Anwendungen

Die Wirkung in UTM-Cluster-Anwendungen ist bei den einzelnen Operanden beschrieben, da die mit KDCLPAP vorgenommenen Änderungen teils Knoten-lokal und teils Clusterglobal wirken.

Wirkungsdauer der Änderungen

Die Wirkungsdauer der Änderungen ist abhängig von der Art der Änderung und ist deshalb bei der Beschreibung der Operanden angegeben.

```
KDCLPAP   { LPAP    ={ lpapname | (lpapname_1,...,lpapname_10) } |
            OSI-LPAP={ osi-lpapname | (osi-lpapname_1,..,osi-lpapname_10) }|
            OSI-CON = osi-conname }
          [ ,ACTION={ CON| (CON,number) | DIS | ACON | (ACON,number) |
                      NACON | QUIET } ]
          [ ,IDLETIME=time_sec ]
          [ ,STATUS={ ON| OFF } ]
```

Für die Administration über Message Queuing müssen Sie KDCLPAPA angeben.

LPAP = (lpapname_1,...,lpapname_10)

Es sollen die Verbindungen zu den Partner-Anwendungen administriert werden, denen die LPAP-Partner *lpapname_1,...,lpapname_10* zugeordnet sind. Für *lpapname_1,...,lpapname_10* sind die logischen Namen von Partner-Anwendungen anzugeben, die mit der KDCDEF-Steueranweisung LPAP für die verteilte Verarbeitung über LU6.1 generiert sind.

Sie können pro Aufruf von KDCLPAP maximal 10 LPAP-Namen angeben, d.h. die Verbindungen zu maximal 10 LPAP-Partnern administrieren. Geben Sie nur einen LPAP-Namen an, können die Klammern entfallen.

OSI-LPAP = (osi-lpapname_1,...,osi-lpapname_10)

Es sollen die Verbindungen zu den Partner-Anwendungen administriert werden, denen die OSI-LPAP-Partner *osi-lpapname_1,...,osi-lpapname_10* zugeordnet sind. Für *osi-lpapname_1,...,osi-lpapname_10* sind die logischen Namen von Partner-Anwendungen anzugeben, die mit der KDCDEF-Steueranweisung OSI-LPAP für die verteilte Verarbeitung über OSI TP generiert sind.

Sie können pro Aufruf von KDCLPAP maximal 10 OSI-LPAP-Namen angeben, d.h. Sie können die Verbindungen zu maximal 10 OSI-LPAP-Partnern administrieren. Geben Sie nur einen OSI-LPAP-Namen an, können die Klammern entfallen.

Handelt es sich bei dem angegebenen OSI-LPAP um das Master-LPAP eines OSI-LPAP-Bündels, so ist nur dioe Angabe von STATUS=ON/OFF sinnvoll.

OSI-CON=osi-conname

Mit KDCLPAP OSI-CON=*osi-conname* wird eine Ersatzverbindung zu einem OSI TP-Partner (OSI-LPAP) aktiviert. Die Ersatzverbindung *osi-con* muss mit einer KDCDEF-Steueranweisung OSI-CON statisch generiert worden sein. Für *osi-conname* ist der in OSI-CON generierte Name anzugeben. Die Namen aller Ersatzverbindungen, die für einen OSI-LPAP-Partner generiert sind, können Sie mit KDCINF OSI-LPAP abfragen.

Zur Zeit der Kommandoeingabe darf keine Verbindung zu dem OSI-LPAP-Partner aufgebaut sein, dem die Ersatzverbindung per Generierung zugeordnet wurde.
Bevor openUTM die Ersatzverbindung aktiviert, deaktiviert openUTM zunächst die zuvor aktive Verbindung zu dem OSI-LPAP-Partner.

Die Ersatzverbindung bleibt aktiv bis zum Ende des Anwendungslaufs bzw. bis zum nächsten Umschalten auf eine Ersatzverbindung zu dem gleichen OSI-LPAP bzw. bis zum Deaktivieren der Verbindung.

Die Eingabe weiterer Operanden ist ohne Wirkung. Die Zuordnung zum OSI-LPAP-Partner erfolgt implizit über *osi-conname*.

ACTION= Mit ACTION können Sie Auf- und Abbau der Verbindungen veranlassen, die durch die Angaben in LPAP oder OSI-LPAP spezifiziert wurden. Sie können festlegen, ob openUTM beim Start der Anwendung automatisch Verbindungen zu den angegebenen Partner Anwendungen aufbauen soll oder nicht.

CON openUTM veranlasst, dass die Verbindungen zu den angegebenen Partner-Anwendungen aufgebaut werden. Zu den in OSI-LPAP angegebenen OSI-TP-Partnern sollen alle in der KDCDEF-Steueranweisung OSI-LPAP generierten parallelen Verbindungen aufgebaut werden.

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Knoten-lokal.

Eine erfolgreiche Kommandoausführung bedeutet nicht, dass die gewünschten Verbindungen aufgebaut sind. Ob der Verbindungsaufbau erfolgreich durchgeführt werden konnte, können Sie mit KDCINF abfragen.

Soll für einen gesperrten LPAP- bzw. OSI-LPAP-Partner eine Verbindung aufgebaut werden, dann muss der Partner vor dem Verbindungsaufbau mit einem eigenen KDCLPAP-Aufruf entsperrt werden.

(CON,number)

Die Angabe von *number* ist nur für Verbindungen zu OSI-LPAP-Partnern sinnvoll. Für einen in LPAP angegebenen LU6.1-Partner wirkt (CON,*number*) wie CON; die Angabe von *number* wird ignoriert.

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Knoten-lokal.

Für OSI-LPAP-Partner ist *number* die Anzahl der parallelen Verbindungen zur Partner-Anwendung, die nach dem KDCLPAP-Aufruf zu jedem einzelnen der angegebenen OSI TP-Partner aufgebaut sein sollen. Die Wirkung des Aufrufs ist somit abhängig von der Angabe für *number*. Folgende Fälle müssen unterschieden werden:

- Ist *number* für einen der in OSI-LPAP angegebenen OSI-LPAP-Partner größer als die Anzahl der derzeit aufgebauten parallelen Verbindungen, dann versucht openUTM soviele Verbindungen aufzubauen, bis *number* Verbindungen zu dem OSI-LPAP-Partner existieren.
  Die maximale Anzahl paralleler Verbindungen zu einem OSI-LPAP-Partner wird bei der KDCDEF-Generierung in der OSI-LPAP-Anweisung festgelegt.
  Überschreitet der Wert von *number* die generierte Maximalzahl für einen der angegebenen OSI-LPAP-Partner, dann baut openUTM für diesen Partner nur die generierte Maximalzahl an Verbindungen auf.
  Eine erfolgreiche Kommandoausführung bedeutet nicht, dass die gewünschten Verbindungen aufgebaut sind. Ob der Verbindungsaufbau erfolgreich durchgeführt werden konnte, können Sie mit KDCINF abfragen.
  Sollen Verbindungen zu einem gesperrten OSI-LPAP-Partner aufgebaut werden, müssen Sie diesen zuvor entsperren.

- Ist *number* kleiner als die Anzahl der derzeit aufgebauten parallelen Verbindungen zu einem OSI-LPAP-Partner, dann baut openUTM soviele Verbindungen zu dem OSI-LPAP-Partner ab, bis nur noch *number* Verbindungen existieren.
  (CON,0) hat die gleiche Wirkung wie ACTION=DIS. openUTM baut für die angegebenen OSI-LPAP-Partner sofort alle Verbindungen ab.

Minimalwert von *number*: 0
Maximalwert von *number*: Anzahl der generierten parallelen Verbindungen.

DIS Die Verbindungen zu den in LPAP angegebenen Partnern werden sofort abgebaut. Alle existierenden parallelen Verbindungen zu den in OSI-LPAP angegebenen Partnern werden abgebaut.

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Knoten-lokal.

> **i** Ein Verbindungsabbau mit DIS wirkt sofort. d.h. alle Verbindungen werden sofort abgebaut. Deshalb ist es möglich, dass Vorgänge abnormal beendet werden. Sie sollten besser ACTION=QUIET verwenden.

ACON (automatic connection)

Beim nächsten Start und bei weiteren Starts der Anwendung soll openUTM die Verbindungen zu den in LPAP bzw. OSI-LPAP angegebenen Partner-Anwendungen automatisch aufbauen. Bei OSI TP-Partnern werden soviele parallele Verbindungen aufgebaut, wie bei der KDCDEF-Generierung in der entsprechenden OSI-LPAP-Anweisung festgelegt.

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Cluster-global.

(ACON,number)

Die Angabe von *number* ist nur für Verbindungen zu OSI-LPAP-Partnern sinnvoll.
Bei allen folgenden Starts der Anwendung soll openUTM die Verbindungen zu den in OSI-LPAP angegebenen Partnern automatisch aufbauen (automatic connection).
Der Wert von *number* legt die Anzahl der parallelen Verbindungen fest, die zu den angegebenen OSI TP-Partnern aufgebaut werden sollen.

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Cluster-global.

Die maximale Anzahl paralleler Verbindungen zu einem Partner wird bei der KDCDEF-Generierung in der OSI-LPAP-Anweisung festgelegt.
Überschreitet der Wert von *number* die generierte Maximalzahl für einen der angegebenen OSI-LPAP-Partner, dann baut openUTM für diesen Partner nur die statisch generierte Anzahl an Verbindungen auf.

Für einen in LPAP angegebenen LU6.1-Partner wirkt (ACON,*number*) wie ACON; die Angabe von *number* wird ignoriert.

Minimalwert von *number*: 0
Maximalwert von *number*: Anzahl der generierten parallelen Verbindungen.

NACON (no automatic connection)
Wenn per Generierung oder per Administration für diese Verbindungen die Eigenschaft ACON eingetragen ist, dann soll sie gelöscht werden, d.h. openUTM soll die Verbindungen zu den in

LPAP oder OSI-LPAP angegebenen Partner-Anwendungen ab dem nächsten Start der Anwendung nicht mehr automatisch aufbauen.
Die Angabe ACTION=NACON wirkt über die Dauer des Anwendungslaufs hinaus.

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Cluster-global.

QUIET Die durch LPAP bzw. OSI-LPAP spezifizierten Verbindungen werden abgebaut. Für OSI-LPAP-Partner werden alle parallelen Verbindungen abgebaut.

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Knoten-lokal.

Bei QUIET werden die Verbindungen erst abgebaut, wenn die zu den angegebenen LPAP- bzw. OSI-LPAP-Partnern generierten Sessions bzw. Associations nicht mehr durch Dialog- oder Asynchron-Aufträge belegt sind. Es werden jedoch keine Dialog-Aufträge für die angegebenen (OSI-)LPAP-Partner mehr angenommen.
Die Eigenschaft QUIET wird durch ACTION=CON zurückgesetzt.

IDLETIME=time_sec

Zeit zur Überwachung des Leerlauf-Zustands (Idle-Zustand) der Sessions bzw. Associations, die für die angegebenen LPAP- bzw. OSI-LPAP-Partner generiert sind.

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Cluster-global.

Die Änderung von IDLETIME wird für eine bestimmte Session bzw. Association wirksam, wenn diese nach Eingabe des Kommandos das nächste Mal den Leerlauf-Zustand erreicht (beim Verbindungsaufbau oder nach der Erledigung eines Auftrags).
Wird dann während der in *time_sec* angegebenen Zeit die Session bzw.
Association von keinem Auftrag belegt, baut openUTM die Verbindung ab. *time_sec* wird in Sekunden angegeben.

IDLETIME=0 bewirkt, dass der Leerlauf-Zustand nicht überwacht wird.

Maximalwert: 32767
Minimalwert: 60
Bei Werten ungleich 0 und kleiner als 60 wird der Wert 60 verwendet.

STATUS= LPAP- bzw. OSI-LPAP-Partner sperren bzw. entsperren.

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Cluster-global.

OFF LPAP- bzw. OSI-LPAP-Partner sperren; openUTM baut zu diesen Partner-Anwendungen keine Verbindung mehr auf, bis die LPAP- bzw. OSI-LPAP-Partner wieder freigegeben werden.

Zum Zeitpunkt des Sperrens eines LPAP- bzw. OSI-LPAP-Partners darf keine logische Verbindung zu der zugehörigen Partner-Anwendung aufgebaut sein.

ON LPAP- bzw. OSI-LPAP-Partner wieder zulassen.
Die Änderung wirkt über die Dauer der laufenden Anwendung hinaus.

Ausgabe von KDCLPAP

Am Administrator-Terminal werden die neuen und alten Eigenschaften (NEW, OLD) ausgegeben.

Bei Eingabe von `KDCLPAP LPAP=...` wird ausgegeben:

```
LPAP          STATUS       CONNECTION          IDLETIME
              NEW  OLD    NEW       OLD       NEW    OLD
lpapname      ON   ON     CON  A Q  CON  A Q  sec    sec
              OFF  OFF    DIS       DIS
                               W         W
```

Bei Eingabe von `KDCLPAP OSI-LPAP=...` wird ausgegeben:

```
OSI-LPAP      STATUS          CONNECTED          IDLETIME       AUTOCON
           NEW      OLD      NEW      OLD       NEW    OLD    NEW     OLD
osi-lpap   ON  Q    ON  Q    number   number    sec    sec    number  number
           OFF      OFF
```

Bei Eingabe von `KDCLPAP OSI-CON=...` wird ausgegeben:

```
OSI-LPAP      OSI-CON
            NEW     OLD
osi-lpap1  osi-con1 osi-con2
```

Erläuterungen zur Ausgabe

AUTOCON	Anzahl der Verbindungen zu dem OSI-LPAP-Partner, die openUTM beim Start der Anwendung automatisch aufbauen soll.
CONNECTED	Anzahl der derzeit zum OSI-LPAP-Partner aufgebauten parallelen Verbindungen.
CONNECTION	1. Spalte: die Verbindung zu dem LPAP-Partner ist aufgebaut (CON), abgebaut (DIS) oder openUTM versucht gerade eine Verbindung aufzubauen (W; waiting for connection). 2. Spalte: A bedeutet: openUTM versucht die Verbindung beim Start der Anwendung automatisch aufzubauen. Q (Quiet) bedeutet: die Verbindung wird abgebaut, für die Partner-Anwendung werden keine Dialog-Aufträge mehr angenommen.
IDLETIME	Zeit zur Überwachung des Leerlauf-Zustands einer Session bzw. Association auf der Verbindung
OSI-CON	der mit der KDCDEF-Steueranweisung OSI-CON generierte Name für die logische Verbindung zur Partner-Anwendung.

osi-con1 ist der Name der Verbindung, die vor dem Umschalten aktiv war,
osi-con2 ist der Name der geschalteten Ersatzverbindung.

STATUS Eine Verbindung zur Partner-Anwendung besteht oder darf aufgebaut (ON), bzw. darf nicht aufgebaut (OFF) werden.
Q (Quiet) bedeutet: Für den OSI-LPAP-Partner werden keine Dialog- Aufträge mehr angenommen, die Verbindung wird abgebaut.

12.8 KDCLSES - Verbindungen für LU6.1-Sessions auf-/abbauen

Mit KDCLSES können Sie für eine Session den Auf- und Abbau einer Transportverbindung veranlassen.

Wirkung in UTM-Cluster-Anwendungen

KDCLSES wirkt in UTM-Cluster-Anwendungen Knoten-lokal.

```
KDCLSES   LSES=lsesname
            ,ACTION={ CON| DIS | QUIET }

          BS2000-Systeme:
          [ ,CON=remote_applname,PRONAM=proname,BCAMAPPL=applname ]

          Unix-, Linux- und Windows-Systeme:
          [ ,CON=remote_applname [,PRONAM=proname ],BCAMAPPL=applname ]
```

Für die Administration über Message Queuing müssen Sie KDCLSESA angeben.

LSES=lsesname

gibt den Namen der zu administrierenden Session an (lokaler Half-Session-Name). Für *lsesname* ist ein Name anzugeben, der mit KC_CREATE_OBJECT Objekttyp KC_LSES oder bei der KDCDEF-Generierung einer LSES-Anweisung zugeordnet wurde.

ACTION= steuert den Auf- und Abbau der Session.

CON Für die Session *lsesname* soll eine Transportverbindung aufgebaut werden.
Mit den Operanden CON, PRONAM und BCAMAPPL können Sie explizit angeben, welche Transportverbindung für die Session aufgebaut werden soll. Geben Sie in CON, PRONAM und BCAMAPPL keine Transportverbindung an, dann versucht openUTM irgendeine der Transportverbindungen aufzubauen, die für den zugehörigen LPAP-Partner generiert ist (KC_CREATE_OBJECT Objekttyp KC_CON oder KDCDEF-Steueranweisung CON).
Kann openUTM die in CON, PRONAM und BCAMAPPL angegebene Verbindung nicht aufbauen, dann versucht openUTM eine andere, für den zugehörigen LPAP-Partner generierte Verbindung aufzubauen.

DIS Die Verbindung, die zur Zeit für die Session aufgebaut ist, wird sofort abgebaut.

QUIET Die Verbindung zur Partner-Anwendung wird abgebaut, wenn die Session nicht mehr durch einen Auftrag belegt ist.

CON=remote_applname, PRONAM=proname, BCAMAPPL=applname

Die Angabe dieser Operanden ist nur bei ACTION=CON sinnvoll.

Mit diesen Operanden können Sie die Transportverbindung, die aufgebaut werden soll, explizit angeben. Die Angaben müssen die Transportverbindung eindeutig identifizieren. Dazu müssen Sie ggf. alle drei Operanden angeben und folgende Angaben machen:

remote_applname
Name der für die Partner-Anwendung generierten Verbindung (remote Half-Session-Name, der dynamisch mit KC_CREATE_OBJECT Objekttyp KC_CON oder mit der KDCDEF-Anweisung CON, der Partner-Anwendung zugeordnet wurde).

proname
Name des Rechners, auf dem die Partner-Anwendung abläuft.
Auf BS2000-Systemen ist dieser Parameter Pflicht.

applname
Anwendungsname der lokalen Anwendung (BCAMAPPL-Name), über den die Verbindung aufgebaut wird. Für *applname* ist der Anwendungsname anzugeben, der dynamisch oder bei der KDCDEF-Generierung in der CON-Anweisung für die Verbindung definiert wurde.

Ausgabe von KDCLSES

Am Administrator-Terminal werden die neuen und alten Eigenschaften (NEW, OLD) der angegebenen Session ausgegeben.

Die Ausgabe ist abhängig davon, ob einem LSES-Objekt ein kurzer oder ein langer Rechnername zugeordnet ist. Bei einem langen Rechnernamen wird die Information zu einem LSES-Objekt in zwei Bildschirmzeilen ausgegeben.

```
LSES     PRONAM   CON              BCAMAPPL  CONNECTION
                                             NEW         OLD
lsesname proname  remote_applname applname  CON|DIS A|Q CON|DIS A|Q
lsesname long.processor.name
                  remote_applname applname  CON|DIS A|Q CON|DIS A|Q
```

Erläuterungen zur Ausgabe

CONNECTION

1. Spalte:
die Verbindung ist aufgebaut (CON) oder abgebaut (DIS).

2. Spalte:
A bedeutet: openUTM versucht, die Verbindung beim Start der Anwendung automatisch aufzubauen.
Q (Quiet) bedeutet: Die Verbindung wird abgebaut, für die Session werden keine Dialog-Aufträge mehr angenommen.

12.9 KDCLTAC - Eigenschaften von LTACs ändern

Mit Hilfe von KDCLTAC können Eigenschaften von LTACs geändert werden. LTACs sind die lokalen TAC-Namen für Services in Partner-Anwendungen bei der verteilten Verarbeitung.

Wirkung in UTM-Cluster-Anwendungen

KDCLTAC wirkt in UTM-Cluster-Anwendungen Cluster-global.

Wirkungsdauer der Änderungen

Die Änderungen wirken höchstens für die Dauer der laufenden Anwendung.

```
KDCLTAC     LTAC={ ltacname | (ltacname_1,...,ltacname_10) }
            [ ,STATUS={ ON| OFF } ]
            [ ,WAITTIME=(accesswait_sec[,replywait_sec]) ]
```

Für die Administration über Message Queuing müssen Sie KDCLTACA angeben.

LTAC=(ltacname_1,...,ltacname_10)

Namen der LTACs, die administriert werden sollen. Für *ltacname_1,...,ltacname_10* sind die Namen von LTACs anzugeben, die dynamisch mit KC_CREATE_OBJECT Objekttyp KC_LTAC erzeugt oder mit der KDCDEF-Steueranweisung LTAC generiert wurden.

Sie können pro Aufruf von KDCLTAC maximal 10 LTAC-Namen angeben.
Geben Sie nur einen LTAC-Namen an, können die Klammern entfallen.

STATUS = LTACs sperren oder Sperren aufheben

OFF die angegebenen LTACs werden gesperrt, es werden keine Aufträge für diesen LTAC mehr angenommen.

ON die Sperre wird aufgehoben; Aufträge für die angegebenen LTACs werden wieder angenommen.

WAITTIME= ersetzt die durch die Generierung bzw. Administration festgelegten Wartezeiten durch die in *accesswait_sec* und *replywait_sec* angegebenen Werte.

accesswait_sec

Zeit in Sekunden, die beim Start eines fernen Services auf das Belegen einer Session (evtl. einschließlich Verbindungsaufbau) bzw. auf den Aufbau einer Association gewartet werden soll.

Eine Wartezeit *accesswait_sec* !=0 bedeutet bei Asynchron-TACs, dass der Auftrag immer in die Message Queue für die Partner-Anwendung eingetragen wird.

Die Wartezeit *accesswait_sec*=0 bedeutet:
Bei Dialog-TACs wird der Vorgang in der lokalen Anwendung sofort mit entsprechendem Returncode fortgesetzt, wenn keine Session bzw. Associa-tion zur Partner-Anwendung frei ist oder die lokale Anwendung „Contention Loser“ ist (siehe KDCDEF-Steueranweisung SESCHA-, LPAP oder OSI-LPAP, Operand CONTWIN).
Bei Asynchron-TACs wird der Asynchron-Auftrag bereits beim FPUT-Aufruf mit einem Returncode

abgewiesen, falls zur Partner-Anwendung keine logische Verbindung besteht. Besteht eine logische Verbindung zur Partner-Anwendung, dann wird die Nachricht in die Ausgabewarteschlange eingetragen.

replywait_sec

Zeit in Sekunden, die openUTM maximal auf die Antwort vom fernen Service der Partner-Anwendung wartet.
Durch Begrenzung der Wartezeit kann gewährleistet werden, dass die Wartezeit für Benutzer am Terminal nicht beliebig lang werden kann.

replywait_sec=0 bedeutet Warten ohne Zeitbegrenzung.

Minimalwert: WAITTIME = (0,0) (Bedeutung siehe oben)
Maximalwert: WAITTIME = (32767,32767)

Ausgabe von KDCLTAC

Am Administrator-Terminal werden die neuen und alten Eigenschaften der angegebenen LTACs ausgegeben.

```
LTAC          STATUS        ACCESSWAIT       REPLYWAIT
             NEW    OLD      NEW     OLD      NEW     OLD
ltacname   ON|OFF ON|OFF    seconds seconds  seconds seconds
```

Erläuterungen zur Ausgabe

LTAC	TAC-Name des fernen Services
STATUS	LTAC gesperrt (OFF) oder nicht (ON)
ACCESSWAIT	Wartezeit auf das Belegen einer Session bzw. Association
REPLYWAIT	Wartezeit auf die Antwort des Service-Programms der Partner-Anwendung

12.10 KDCLTERM - Eigenschaften von LTERM-Partnern ändern

Mit KDCLTERM können Sie die Eigenschaften der LTERM-Partner von Clients, Druckern und LTERM-Pools ändern. Sie können die LTERM-Partner sperren, Sperren aufheben und die Verbindung zu den Clients und Druckern auf- und abbauen.

Wirkung in UTM-Cluster-Anwendungen

Die Wirkung in UTM-Cluster-Anwendungen ist bei den einzelnen Operanden beschrieben, da die mit KDCLTERM vorgenommenen Änderungen teils Knoten-lokal und teils Cluster-global wirken.

Wirkungsdauer der Änderungen

Alle Änderungen bleiben über das Anwendungsende hinaus erhalten.

```
KDCLTERM    LTERM={ ltermname | (ltermname_1,ltermname_2,...,ltermname_10) }
            [ ,ACTION={ CON| DIS } ]
            [ ,STATUS={ ON| OFF } ]
            [ ,PRIMARY=primary-lterm]
```

Für die Administration über Message Queuing müssen Sie KDCLTRMA angeben.

LTERM=(ltermname_1,...,ltermname_10)

Namen der LTERM-Partner, die administriert werden sollen.
Sie können pro Aufruf von KDCLTERM maximal 10 LTERM-Partner angeben. Geben Sie nur einen LTERM-Partner *ltermname* an, können die Klammern entfallen.

Für *ltermname_1,...,ltermname_10* können Sie auch die Namen von LTERM-Partnern angeben, die einem LTERM-Pool zugeordnet sind. Dabei ist der vollständige Name dieser LTERM-Partner anzugeben, also LTERM-Präfix des LTERM-Pools und die laufende Nummer.
Sie können jedoch die LTERM-Partner, die zu LTERM-Pools gehören, nicht sperren und auch keine Verbindung zu ihnen aufbauen, d.h. für sie ist nur die Angabe von ACTION=DIS (Verbindungsabbau) erlaubt.

ACTION= Verbindungen zu den LTERM-Partnern auf- oder abbauen

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Knoten-lokal.

CON Es wird der Aufbau der Verbindungen zu den angegebenen LTERM-Partnern veranlasst.
Für LTERM-Partner eines LTERM-Pools ist ACTION=CON nicht erlaubt.

In UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen wirkt die Funktion nur für TS-Anwendungen und Drucker.

DIS Die Verbindungen zu den angegebenen LTERM-Partnern werden abgebaut. Der Verbindungsabbau ist sofort wirksam, d.h. offene Vorgänge können nicht beendet werden.

STATUS= LTERM-Partner sperren oder wieder zulassen.

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Cluster-global.

ON LTERM-Partner wieder zulassen

OFF die angegebenen LTERM-Partner sperren.

Die Sperre wirkt wie folgt:

- Ein Verbindungswunsch wird ausgeführt. Die Verbindung wird aufgebaut und es wird folgende Meldung ausgegeben:

  ```
  K027 LTERM-Partner ltermname gesperrt - Administrator verständigen oder KDCOFF eingeben
  ```

- Eine bestehende Verbindung bleibt erhalten. Jede Eingabe mit Ausnahme von KDCOFF wird mit folgender Meldung quittiert:

  ```
  K027 LTERM-Partner ltermname gesperrt - Administrator verständigen oder KDCOFF eingeben
  ```

  Die Sperre wirkt erst, wenn auf dieser Verbindung ein Sicherungspunkt (Transaktionsende) erreicht ist.

KDCOFF BUT wirkt bei gesperrtem LTERM-Partner wie KDCOFF.

LTERM-Partner eines LTERM-Pools können nicht gesperrt werden.

PRIMARY=primary-lterm

Name eines normalen LTERMs oder eines Primary-LTERMs einer LTERM-Gruppe (siehe openUTM-Handbuch „Anwendungen generieren“).

Der Operand PRIMARY ist nur in stand-alone UTM-Anwendungen erlaubt.

LTERM muss ein Alias-LTERM einer LTERM-Gruppe sein. Durch Angabe von PRIMARY= wird es zu einem Alias-LTERM der LTERM-Gruppe mit dem Primary-LTERMs *primary-lterm*.

Ist *primary-lterm* bereits das Primary-LTERM einer LTERM-Gruppe, wird das LTERM dieser Gruppe zugeordnet. Waren *primary-lterm* bisher keine Alias-LTERMs zugeordnet, entsteht jetzt eine neue LTERM-Gruppe mit *primary-lterm* als Primary-LTERM und *ltermname (LTERM=)* als Alias-LTERM.

Ist *primary-lterm* ein normales LTERM, muss es folgende Bedingungen erfüllen:

- Ihm muss ein PTERM mit PTYPE APPLI oder SOCKET zugeordnet sein.
- Es darf kein Slave-LTERM eines LTERM-Bündels sein.
- Es muss mit USAGE=D generiert sein.

Ausgabe von KDCLTERM

Am Administrator-Terminal werden die neuen und alten Eigenschaften (NEW, OLD) des LTERM-Partners angezeigt. Die Anzeige 'POOL LTERM' besagt, dass der Client über einen LTERM-Pool angeschlossen ist.

```
LTERM      STATUS            CONNECTION
           NEW      OLD      NEW        OLD
ltermname  ON|OFF   ON|OFF   CON|DIS|W  CON|DIS|W [POOL LTERM]
```

Erläuterung der Ausgabe

LTERM	Name des LTERM-Partners
STATUS	Der LTERM-Partner ist gesperrt (OFF) oder nicht gesperrt (ON).
CONNECTION	Die Verbindung zu dem LTERM-Partner ist aufgebaut (CON), abgebaut (DIS) oder openUTM versucht gerade eine Verbindung aufzubauen (W; waiting for connection).

12.11 KDCMUX - Eigenschaften von Multiplex-Anschlüssen ändern (BS2000-Systeme)

Mit KDCMUX können Sie die Eigenschaften von Multiplex-Anschlüssen wie folgt ändern:

- Multiplex-Anschlüsse sperren bzw. entsperren,
- Multiplex-Anschlüsse aktivieren oder deaktivieren,
- Angaben zum Verbindungsaufbau beim Start der Anwendung machen.

Besonderheiten beim Verbindungsaufbau

Mit KDCMUX ...,ACTION=CON wird der Verbindungsaufbau nur veranlasst. Eine erfolgreiche Kommandoausführung bedeutet also nicht, dass die Verbindungen aufgebaut sind bzw. wirklich aufgebaut werden können (z.B. kein Verbindungsaufbau wegen eines Fehlers im Transportsystem). Ob openUTM eine Verbindung tatsächlich aufbauen konnte, sollten Sie deshalb mit KDCINF kontrollieren. Zum Beispiel durch:

```
KDCINF MUX,LIST=(muxname_1,muxname_2,...,muxname_10)
```

Soll für einen gesperrten Multiplexanschluss (STATUS=OFF) eine Verbindung aufgebaut werden, dann müssen Sie KDCMUX zweimal aufrufen.

1. KDCMUX zum Entsperren des Multiplexanschlusses, z.B.:
   `KDCMUX MUX=muxname,STATUS=ON`
2. KDCMUX-Aufruf, der veranlasst, dass die Verbindung aufgebaut wird, z.B.:
   `KDCMUX MUX=muxname,ACTION=CON`

Wirkungsdauer der Änderungen

Die Wirkungsdauer der Änderungen ist abhängig von der Art der Änderung und ist deshalb bei der Beschreibung der Operanden angegeben.

```
KDCMUX      MUX={ muxname | (muxname_1,muxname_2,...,muxname_10) }
            [ ,ACTION={ CON| DIS | ACON | NACON } ]
            [ ,BCAMAPPL=applname ]
            [ ,MAXSES=number_sessions ]
            ,PRONAM=proname
            [ ,STATUS={ ON| OFF } ]
```

Für die Administration über Message Queuing müssen Sie KDCMUXA angeben.

MUX=(muxname_1,muxname_2,...,muxname_10)

Namen der Multiplex-Anschlüsse, die administriert werden sollen.
Für *muxname_1,...,muxname_10* müssen Namen angegeben werden, die bei der KDCDEF-Generierung mit MUX-Anweisungen definiert worden sind.
Sie können pro Aufruf von KDCMUX maximal 10 Namen angeben. Geben Sie nur einen Namen an, können die Klammern entfallen.

ACTION= Mit ACTION können Sie den Auf- und Abbau der Verbindungen zu den angegebenen Multiplex-Anschlüssen veranlassen. Sie können angeben, ob openUTM bei folgenden Starts der Anwendung automatisch Verbindungen zu den angegebenen Multiplex-Anschlüssen aufbauen soll oder nicht.

CON (connection)
openUTM veranlasst den Aufbau der Verbindungen zu den angegebenen Multiplex-Anschlüssen.

Ob der Verbindungsaufbau erfolgreich durchgeführt werden konnte, können Sie mit KDCINF abfragen.

DIS (disconnection)
Die Verbindungen zu den angegebenen Multiplex-Anschlüssen werden abgebaut. Der Verbindungsabbau ist sofort wirksam, auch offene Vorgänge können nicht beendet werden.

ACON (automatic connection)
Bei den folgenden Starts der Anwendung soll openUTM die Multiplex-Anschlüsse automatisch aktivieren, d.h. die Verbindungen automatisch aufbauen.

ACTION=ACON wirkt solange, bis der automatische Verbindungsaufbau durch die Administration explizit zurückgesetzt wird (Aktion NACON).

NACON (no automatic connection)
Wenn per Generierung oder per Administration für die angegebenen Multiplex-Anschlüsse die Eigenschaft ACON eingetragen ist, dann soll sie gelöscht werden. D.h. die Verbindungen zu den angegebenen Multiplex-Anschlüssen werden bei den folgenden Starts der Anwendung nicht mehr automatisch aufgebaut.

BCAMAPPL=applname

Anwendungsname der lokalen Anwendung, über den die Verbindungen zu den Multiplex-Anschlüssen aufgebaut werden. Für *applname* ist der Anwendungsname anzugeben, der den Multiplex-Anschlüssen bei der KDCDEF-Generierung in den MUX-Anweisungen zugeordnet wurde, d.h der Name, den der Nachrichtenverteiler beim Verbindungsaufbau zur UTM-Anwendung angeben muss.

Standardwert:
Geben Sie BCAMAPPL nicht an, dann wird der in der KDCDEF-Steueranweisung MAX im Operanden APPLINAME generierte Name angenommen.

MAXSES=number_sessions

number_sessions legt die maximale Anzahl der Terminals fest, die über jeden dieser Multiplex-Anschlüsse gleichzeitig angeschlossen werden können.

Die Änderung wirkt nur für den aktuellen Anwendungslauf.

Minimalwert: 1
Maximalwert: 65000

PRONAM=proname

Name des Prozessors, auf dem der Nachrichtenverteiler abläuft, der dem Multiplexanschluss zugeordnet ist.

STATUS= Die angegebenen Multiplex-Anschlüsse werden gesperrt oder wieder freigegeben.

ON Sperre der Multiplex-Anschlüsse aufheben (wirkt sofort).

OFF Multiplex-Anschlüsse sperren. Zum Zeitpunkt des Sperrens darf keine Verbindung zu einem der angegebenen Multiplex-Anschlüsse aufgebaut sein.
Existiert eine Verbindung, dann setzt openUTM die Sperre nicht. In der Ausgabe von KDCMUX wird für STATUS NEW und für STATUS OLD der Wert ON ausgegeben.

Ausgabe von KDCMUX

Am Administrator-Terminal werden die neuen und alten (NEW, OLD) Eigenschaften der administrierten Multiplex-Anschlüsse angezeigt.

```
MUX      PRONAM    BCAMAPPL  STATUS          CONNECTION      MAXSES
                             NEW     OLD     NEW     OLD     NEW     OLD
muxname  proname   applname  ON|OFF  ON|OFF  AC|D|W  AC|D|W  number  number
```

Erläuterungen zur Ausgabe

MUX Name des Multiplexanschlusses

PRONAM Name des Prozessors, auf dem der Nachrichtenverteiler abläuft

BCAMAPPL Name der UTM-Anwendung, der dem Multiplexanschluss bei der KDCDEF-Generierung zugeordnet wurde.

STATUS Der Multiplexanschluss ist gesperrt (OFF) oder nicht gesperrt (ON).

CONNECTION 1. Spalte:
Der Multiplexanschluss ist mit der Anwendung verbunden (C) oder nicht verbunden (D) oder openUTM versucht gerade eine Verbindung zu dem Multiplexanschluss aufzubauen (W; waiting for connection).

2. Spalte:
openUTM versucht bei jedem Start der Anwendung, die Verbindung automatisch aufzubauen (A) oder nicht (keine Angabe).

MAXSES Maximale Anzahl der Terminals, die gleichzeitig über diesen Multiplexanschluss an die Anwendung angeschlossen werden können.

12.12 KDCPOOL - LTERM-Pools administrieren

Mit KDCPOOL können Sie die Anzahl der für einen LTERM-Pool zugelassenen bzw. gesperrten Clients neu festlegen.

Wirkung in UTM-Cluster-Anwendungen

KDCPOOL wirkt in UTM-Cluster-Anwendungen Cluster-global.

Wirkungsdauer der Änderung

Die Änderung bleibt über das Beenden der Anwendung hinaus erhalten.

```
KDCPOOL     LTERM=ltermprefix
            [ ,STATUS=( { ON| OFF }, number_clients ) ]
```

Für die Administration über Message Queuing müssen Sie KDCPOOLA angeben.

LTERM=ltermprefix

LTERM-Präfix des LTERM-Pools, wie in der KDCDEF-Steueranweisung TPOOL generiert. Durch die Angabe von *ltermprefix* wird der zu administrierende LTERM-Pool eindeutig identifiziert.

STATUS= legt die Anzahl der Clients fest, die sich gleichzeitig über den LTERM-Pool anschließen können. Der Maximalwert der Clients, die sich gleichzeitig über den LTERM-Pool anschließen können, wird bei der KDCDEF-Generierung festgelegt (NUMBER in der Steueranweisung TPOOL). Per Administration können Sie diese Zahl herabsetzen oder eine zuvor herabgesetzte Anzahl bis zu der Maximalzahl wieder heraufsetzen.

(ON,number_clients)

number_clients legt die Anzahl der zugelassenen LTERM-Partner des LTERM-Pools fest.

(OFF,number_clients)

number_clients legt die Anzahl der gesperrten LTERM-Partner des LTERM-Pools fest, d.h. die bei der KDCDEF-Generierung angegebene maximale Anzahl der LTERM-Partner wird um *number_clients* herabgesetzt. Die Sperre von LTERM-Partnern eines LTERM-Pools wirkt sich wie folgt aus:

- ein Verbindungsaufbauwunsch eines Client wird von openUTM abgelehnt, sobald die für den LTERM-Pool zugelassene Anzahl der LTERM-Partner durch andere Clients belegt ist.
- bestehen zur Zeit der Kommandobearbeitung mehr Verbindungen zu dem LTERM-Pool als LTERM-Partner zugelassen werden, dann bleiben zunächst alle bestehende Verbindungen erhalten.
  Die Sperre wird erst nach dem Verbindungsabbau wirksam, wenn von einem Client ein neuer Verbindungsaufbauwunsch kommt.
  Meldet sich ein Terminal-Benutzer mit KDCOFF BUT ab, dann kann er sich mit KDCSIGN wieder anmelden, auch wenn zu diesem Zeitpunkt noch mehr LTERM-Partner des LTERM-Pools belegt sind als zugelassen.

Minimalwert von *number_clients*: 0

Maximalwert von *number_clients*:
die bei der KDCDEF-Generierung angegebene Maximalzahl von Clients, die sich gleichzeitig über diesen LTERM-Pool anschließen können. Wenn beim Zulassen von Clients (ON,*number_clients*) *number_clients* größer als der Maximalwert ist, setzt openUTM den Wert von *number_clients* automatisch auf den Maximalwert herab.

Ausgabe von KDCPOOL

Am Administrator-Terminal werden die neue und alte Zahl der für den LTERM-Pool zugelassenen Clients in folgender Form ausgegeben.

Die Ausgabe ist abhängig davon, ob einem LTERM-Pool ein kurzer oder ein langer Rechnername zugeordnet ist. Bei einem langen Rechnernamen wird die Information zu einem LTERM-Pool in zwei Bildschirmzeilen ausgegeben.

```
POOL        PRONAM   BCAMAPPL  PTYPE  STA=ON
                                      NEW    OLD
ltermprefix proname  applname  ptype  number number
ltermprefix long.processor.name
                     applname  ptype  number number
```

Erläuterungen zur Ausgabe

POOL	LTERM-Präfix, das für den LTERM-Pool generiert wurde.
PRONAM	Name des Rechners, dem der LTERM-Pool zugeordnet wurde.
BCAMAPPL	Name der UTM-Anwendung, der dem LTERM-Pool bei der KDCDEF-Generierung zugeordnet wurde.
PTYPE	Typ der Clients, die sich über den LTERM-Pool anschließen können.
STA=ON	Anzahl der LTERM-Partner, die vor der Kommandobearbeitung für den LTERM-Pool zugelassen waren (OLD), und Anzahl der LTERM-Partner, die aktuell für den LTERM-Pool zugelassen sind (NEW).

12.13 KDCPROG - Lademodule/Shared Objects/DLLs austauschen

Mit KDCPROG können Sie Lademodule einer UTM-Anwendung unter einem BS2000-System mit Hilfe der BLS-Schnittstelle austauschen bzw. Shared Objects einer UTM-Anwendung auf Unix- und Linux-Systemen, wenn Sie mit der Funktion „Programmaustausch mit Shared Objects“ arbeiten. Sehen Sie dazu auch das openUTM-Handbuch „Anwendungen generieren“ sowie das betreffende openUTM-Handbuch „Einsatz von UTM-Anwendungen“.

Auf Windows-Systemen werden Shared Objects mit Hilfe von DLLs realisiert. Besonderheiten zu deren Handhabung sind im openUTM-Handbuch „Einsatz von UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen“ beschrieben.

Voraussetzungen für den Programmaustausch mit KDCPROG

Sie können Teile eines Anwendungsprogramms austauschen bzw. nachladen, die die folgenden Bedingungen erfüllen:

- Die Programmteile, die ausgetauscht werden sollen, wurden als eigenständige Lademodule/Shared Objects /DLLs generiert.
- Jedes auszutauschende Lademodul bzw. Shared Object/DLL wurde mit einer LOAD-MODULE-Anweisung (BS2000-Systeme) bzw. SHARED-OBJECT-Anweisung (Unix-, Linux- und Windows-Systeme) statisch generiert.
- Die Lademodule/Shared Objects/DLLs wurden nicht statisch zum Anwendungsprogramm hinzugebunden.
- Auf BS2000-Systemen wurden die Lademodule , die ausgetauscht werden sollen, weder in den Systemspeicher (Klasse-4-Speicher) noch in einen Anwendungsglobalen Common Memory Pool (generiert mit SCOPE=GLOBAL) geladen.

Damit openUTM das Kommando bearbeiten kann, muss ein Lademodul, Shared Object bzw. DLL mit dem angegebenen Namen und der Version *version* in der Programmbibliothek bzw. im Directory vorhanden sein, die /das ihm durch die KDCDEF-Generierung zugeordnet wurde:

- LOAD-MODULE-Anweisung, Operand LIB (BS2000-Systeme) bzw.
- SHARED-OBJECT-Anweisung, Operand DIRECTORY (Unix-, Linux- und Windows-Systeme).

BS2000-Systeme

Ist in der Programmbibliothek kein Lademodul mit angegebenem Namen und Version vorhanden, dann wird das Administrationskommando abgewiesen und das bisher geladene Lademodul bleibt geladen. Zusätzlich wird die Meldung K234 ausgegeben.

Unix-, Linux- und Windows-Systeme

Ist im angegebenen Directory kein Shared Object/DLL mit der Version *version* vorhanden, so wird nur das bisher geladene Shared Object/DLL entladen und eine Meldung ausgegeben.

Durch einen erneuten Aufruf von KDCPROG können Sie dann das Lademodul bzw. Shared Object/DLL laden, indem Sie für *version* eine in der Bibliothek vorhandene Version des Lademoduls/Shared Objects angeben oder das fehlende Lademodul bzw. Shared Object/DLL mit der angegebenen Version in die Programmbibliothek /Programmdateiverzeichnis stellen.

Ablauf des Programmaustausches

Wie der Programmaustausch durchgeführt wird, ist abhängig vom generierten Lademodus des Lademoduls/Shared Objects/DLL.

Den Lademodus generieren Sie in der LOAD-MODULE-Anweisung (BS2000-Systeme) bzw. in der SHARED-OBJECT-Anweisung (Unix-, Linux- und Windows-Systeme) jeweils im Operanden LOAD-MODE.

- LOAD-MODE=STARTUP
  (das Lademodul/Shared Object/DLL wird beim Start der Anwendung als eigenständige Einheit geladen)
  Der Austausch wird für jeden Prozess spätestens nach der Bearbeitung des nächsten Auftrags ausgeführt, ohne dass das laufende Anwendungsprogramm beendet wird. Mehrere Prozesse der Anwendung können gleichzeitig austauschen. Bis der Programmaustausch von allen Prozessen der Anwendung abgeschlossen wurde, dürfen Sie keinen weiteren Austausch mit KDCPROG veranlassen.

  Der Programmaustausch wird durch den Aufruf von KDCPROG nur veranlasst. Der eigentliche Programmaustausch kann sich über einen längeren Zeitraum erstrecken. openUTM informiert mit Meldungen auf SYSOUT und SYSLOG (BS2000-Systeme) bzw. *stdout* und *stderr* (Unix-, Linux- und Windows-Systeme) über Erfolg oder Misserfolg des Programmaustausches.
- LOAD-MODE=ONCALL
  (wird geladen, wenn ein Teilprogramm aus dem Lademodul/Shared Object/DLL erstmals aufgerufen wird)
  Der Austausch wird für jeden Prozess erst dann durchgeführt, wenn das nächste Mal in diesem Prozess ein Teilprogramm aus diesem Lademodul/Shared Object aufgerufen wird. Zu einer Zeit können Sie mehrere Prozesse der Anwendung gleichzeitig austauschen.
- LOAD-MODE=(POOL, POOL/STARTUP, POOL/ONCALL, nur auf BS2000-Systemen) (die Public Slice des Lademoduls wird in einen Common Memory Pool geladen)
  In diesem Fall wird durch den Aufruf von KDCPROG der Programmaustausch **nicht** angestoßen. KDCPROG bewirkt nur ein Vormerken der neuen Version. Erst beim folgenden Austausch des gesamten Anwendungsprogramms mit KDCAPPL PROG=NEW wird die neue Version des Lademoduls in den Common Memory Pool geladen.
  In diesem Fall kann unmittelbar nach dem Aufruf von KDCPROG ein weiterer KDCPROG-Aufruf folgen.
  Sie können durch mehrere KDCPROG-Aufrufe mehrere Lademodule vormerken, die dann beim folgenden KDCAPPL PROG=NEW ausgetauscht werden. Folgt im selben Anwendungslauf kein KDCAPPL PROG=NEW, dann werden die vorgemerkten Versionen beim folgenden Anwendungsstart ausgetauscht.

Wirkung in UTM-Cluster-Anwendungen auf Unix-, Linux- und Windows-Systemen:

KDCPROG wirkt in UTM-Cluster-Anwendungen Cluster-global. Ein Programmaustausch wird in allen laufenden Knoten-Anwendungen durchgeführt.

Wirkungsdauer des Programmaustausches

Die Änderung wirkt über das Ende der Anwendung hinaus.

```
KDCPROG     VERSION={ version | *HIGHEST-EXISTING | *UPPER-LIMIT }
```

BS2000-Systeme:

```
            , LOAD-MODULE=lmodname
```

Unix-, Linux- und Windows-Systeme:

```
            , SHARED-OBJECT=shared-object-name
```

Für die Administration über Message Queuing müssen Sie KDCPROGA angeben.

VERSION=version

Version des Lademoduls/Shared Objects/DLL, die geladen werden soll. Der Wert von *version* darf maximal 24 Zeichen lang sein.

BS2000-Systeme

- In UTM-Anwendungen auf BS2000-Systemen müssen Sie die zu ladende Version des Lademoduls immer angeben.
- Für Lademodule, die mit LOAD-MODE=STARTUP generiert sind, dürfen die Versionsnummern von altem und neuem Lademodul übereinstimmen.
- Für Lademodule, die mit LOAD-MODE=ONCALL generiert sind oder die ganz oder teilweise in einem Common Memory Pool liegen, muss sich die neue Versionsnummer von der alten Versionsnummer unterscheiden.
- Beim Anstoßen des Austausches muss in der Bibliothek, die dem Lademodul bei der KDCDEF-Generierung zugeordnet wurde (siehe auch KDCDEF-Anweisung LOAD-MODULE...,LIB=), ein Element mit dem Namen *lmodname* und der in *version* angegebenen Version vorhanden sein.
- Wird VERSION=*HIGHEST-EXISTING angegeben, dann wird die höchste in der Bibliothek vorhandene Version dieses Moduls ermittelt und geladen.
- Wird VERSION=@ bzw. *UPPER-LIMIT angegeben, dann wird das Lademodul geladen, das als letztes ohne explizite Versionsangabe in diese PLAM-Bibliothek gebracht wurde. Wird bei LMS mit expliziten Versionen gearbeitet, kann @ bzw. *UPPER-LIMIT als Version eines Lademoduls nicht verwendet werden.
- Ist ein Lademodul mit LOAD-MODE=POOL, (POOL,STARTUP) oder (POOL,ONCALL) sowie mit Version *HIGHEST-EXISTING generiert, dann darf bei Version **nur** *HIGHEST-EXISTING angegeben werden. Ein solcher Modul kann nur über einen Anwendungsaustausch neu geladen werden; dabei wird für einen so generierten Modul immer die höchste verfügbare Version geladen.
- Sie können keine Lademodule austauschen, die statisch zum Anwendungsprogramm gebunden wurden (Lademodus STATIC).

Lademodule mit Lademodus STARTUP, die TCB-Entries enthalten, dürfen ebenfalls nicht ausgetauscht werden.

LOAD-MODULE=lmodname (nur auf BS2000-Systemen)

Name des Lademoduls, das ausgetauscht werden soll. Dies kann der Name eines OM oder eines LLM sein. Das Lademodul muss (mit diesem Namen) bei der KDCDEF-Generierung mit einer LOAD-MODULE-Anweisung konfiguriert worden sein. Sie dürfen pro Aufruf von KDCPROG nur einen Namen angeben.
Der Name darf maximal 32 Zeichen lang sein.

SHARED-OBJECT=shared-object-name (nur auf Unix-, Linux- und Windows-Systemen)

Name des Shared Objects/DLL, das ausgetauscht werden soll. Der Name muss mit einer SHARED-OBJECT-Anweisung generiert worden sein. Sie dürfen pro Aufruf von KDCPROG nur einen Namen angeben.
Der Name darf maximal 32 Zeichen lang sein.

Ausgabe von KDCPROG

Nach dem Absetzen eines KDCPROG-Aufrufs werden am Administrator-Terminal folgende Informationen ausgegeben:

Ausgabe für UTM-Anwendungen auf BS2000-Systemen

```
LOAD-MODULE         lmodname
VERSION (GENERATED) generated element version
VERSION (PREVIOUS)  old element version
VERSION (CURRENT)   new element version
LIBRARY             name of program library
LOAD MODE           STARTUP | ONCALL | POOL | POOL/STARTUP | POOL/ONCALL
```

Ausgabe für UTM-Anwendung auf Unix-, Linux- und Windows-Systemen

```
SHARED-OBJECT       shared object name
VERSION (PREVIOUS)  old  version
VERSION (CURRENT)   new  version
DIRECTORY           name of program directory
LOAD MODE           STARTUP | ONCALL
```

Erläuterungen zur Ausgabe

LOAD-MODULE

Name des Lademoduls auf BS2000-Systemen

SHARED-OBJECT

Name des Shared Objects/DLL auf Unix-, Linux- und Windows-Systemen

VERSION (GENERATED)

generierte Version des Lademoduls auf BS2000-Systemen

VERSION (PREVIOUS)

bisher geladene Version des Lademoduls/Shared Objects/DLL

VERSION (CURRENT)

Version des Lademoduls/Shared Objects/DLL, das geladen werden soll

LIBRARY Name der Programmbibliothek, aus der das Lademodul (BS2000-Systeme) geladen wird.

DIRECTORY Name des Verzeichnisses, aus dem das Shared Object/DLL (Unix-, Linux- und Windows-Systeme) nachgeladen wird

LOAD MODE Lademodus des Lademoduls/Shared Objects/DLL:

STARTUP
Das Lademodul/Shared Object/DLL wird beim Start der Anwendung als eigenständige Einheit geladen.

ONCALL
Das Lademodul/Shared Object/DLL wird geladen, wenn ein Teilprogramm aus dem Lademodul erstmals aufgerufen wird.

Nur auf BS2000-Systemen:

POOL
Das Lademodul wird beim Start der Anwendung in den Common Memory Pool geladen. Das Lademodul enthält keine Private Slice.

POOL/STARTUP
Die Public Slice des Lademoduls wird beim Start der Anwendung in den Common Memory Pool geladen. Die zu dem Lademodul gehörende Private Slice wird anschließend in den prozesslokalen Speicher geladen.

POOL/ONCALL
Die Public Slice des Lademoduls wird beim Start der Anwendung in den Common Memory Pool geladen. Die zu dem Lademodul gehörende Private Slice wird anschließend in den prozesslokalen Speicher geladen, wenn das erste Teilprogramm aufgerufen wird, das diesem Lademodul zugeordnet ist

Meldungen zum Programmaustausch mit KDCPROG

Wenn der Austausch von Anwendungsteilen, die mit einem Lademodus von STARTUP (bei BS2000-Systemen auch POOL) generiert wurden, abgeschlossen ist, wird folgende Meldung nach SYSOUT und SYSLOG (BS2000-Systeme) bzw. *stdout* und *stderr* (Unix-, Linux- und Windows-Systeme) ausgegeben:

```
K074 Programmaustausch abgeschlossen;..
```

Wenn beim Austausch von Anwendungsteilen, die mit Lademodus STARTUP (bei BS2000-Systemen auch POOL) generiert wurden, Fehler auftreten, wird folgende Meldung nach SYSLST und SYSLOG (BS2000-Systeme) bzw. *stdout* und *stderr* (Unix-, Linux- und Windows-Systeme) ausgegeben:

```
K075 Programmaustausch von Task/Prozess .. abgebrochen ...
```

Treten auf BS2000-Systemen beim Austausch Fehler auf, dann wird für jeden Prozess die Meldung K078 mit der Fehlerursache nach SYSOUT ausgegeben.

12.14 KDCPTERM - Eigenschaften von Clients und Druckern ändern

Mit KDCPTERM können Sie die Eigenschaften von Clients und Druckern ändern.

Folgende Aktionen können Sie durchführen:

- Clients und Drucker sperren bzw. wieder zulassen.
- Logische Verbindungen zu Clients und Druckern auf- und abbauen. Insbesondere können Sie die Verbindungen zu den einzelnen Druckern eines Bündels auf- bzw. abbauen.
- Den automatischen Aufbau der Verbindungen zu Clients und Druckern beim Start der Anwendung veranlassen oder verhindern.
- Nur auf BS2000-Systemen: Sind Terminals über einen Multiplexanschluss mit der Anwendung verbunden, dann können Sie Sessions freigeben, die sich im Zustand DISCONNECT-PENDING befinden.

Besonderheiten beim Verbindungsaufbau und -abbau

Mit KDCPTERM können Sie den sofortigen Verbindungsaufbau bzw. den automatischen Verbindungsaufbau bei jedem folgenden Anwendungsstart zu folgenden Objekten veranlassen (ACTION=CON oder ACON):

- auf BS2000-Systemen zu Druckern, Terminals und Transportsystem-Anwendungen vom Typ APPLI oder SOCKET.
  Anforderungen von Verbindungen zu UTM-Clients mit Trägersystem UPIC (PTYPE=UPIC-R) werden abgewiesen.
  Die Initiative zum Verbindungsaufbau liegt immer beim UTM-Client.
- auf Unix-, Linux- und Windows-Systemen zu Druckern (PTYPE=PRINTER) und Transportsystem-Anwendungen vom Typ PTYPE=APPLI oder SOCKET.
  Anforderungen von Verbindungen zu UTM-Clients mit Trägersystem UPIC (PTYPE=UPIC-R/L) und zu Terminals (PTYPE=TTY) werden abgewiesen. Der Aufbau von Verbindungen zu diesen Clients kann nur von den Clients selbst initiiert werden.

Zu Clients, die sich über einen LTERM-Pool an die Anwendung anschließen, kann mit KDCPTERM keine Verbindung aufgebaut werden.

Wird der Verbindungsaufbau zu einem Client angefordert, für den die Aktionen CON und ACON nicht erlaubt sind, wird der KDCPTERM-Aufruf abgewiesen.

Bei einem erfolgreichen Aufruf von KDCPTERM mit Aktion CON veranlasst openUTM den Verbindungsaufbau zu den angegebenen Clients und Druckern. Eine erfolgreiche Kommandoausführung bedeutet jedoch nicht, dass die Verbindungen aufgebaut sind bzw. wirklich aufgebaut werden können. Ob die Verbindungen tatsächlich zustande kommen, muss explizit abgefragt werden (z.B. mit KDCINF).

Ein Verbindungsabbau mit ACTION=DIS bewirkt, dass die Verbindung zu dem Client oder Drucker sofort abgebaut wird. Auch offene Vorgänge können nicht beendet werden.

Besonderheiten beim Sperren von Clients oder Druckern

Eine Sperre wirkt wie folgt:

- Jeder Verbindungswunsch eines Client wird abgelehnt.
- Eine bestehende Verbindung bleibt erhalten.
  Die Sperre wirkt erst beim nächsten Versuch des Client, eine logische Verbindung aufzubauen.

- Die Anforderung einer Verbindung zu einem gesperrten Client oder Drucker wird abgelehnt.

Asynchrone Nachrichten an gesperrte Clients oder Drucker werden auf der KDCFILE zwischengespeichert und können zu Betriebsmittelengpass führen!

Wirkung in UTM-Cluster-Anwendungen

Die Wirkung in UTM-Cluster-Anwendungen ist bei den einzelnen Operanden beschrieben, da die mit KDCPTERM vorgenommenen Änderungen teils Knoten-lokal und teils Cluster-global wirken.

Wirkungsdauer der Änderungen

Die Wirkungsdauer der Änderungen ist abhängig von der Art der Änderung und ist deshalb bei der Beschreibung der Operanden angegeben.

```
KDCPTERM    PTERM={ ptermname | (ptermname_1,ptermname_2,...,ptermname_10) }
            [ ,BCAMAPPL=applname ]
            [ ,STATUS={ ON| OFF } ]
           BS2000-Systeme:
            ,PRONAM=proname
            [ ,ACTION={ CON| DIS | REL | ACON | NACON } ]
           Unix-, Linux- und Windows-Systeme:
            [ ,PRONAM=proname ]
            [ ,ACTION={ CON| DIS | ACON | NACON } ]
```

Für die Administration über Message Queuing müssen Sie das Administrationskommando KDCPTRMA angeben.

PTERM=(ptermname_1,...,ptermname_10)

Namen der zu administrierenden Clients und Drucker. Sie können pro KDCPTERM-Aufruf maximal 10 Namen angeben.
Wird nur ein Name angegeben, können die Klammern entfallen.

Alle Namen der Liste müssen zu Clients und Druckern gehören, die sich an demselben Rechner befinden.

BCAMAPPL=applname

nur sinnvoll bei Client-Anwendungen vom Typ PTYPE=APPLI/SOCKET oder PTYPE=UPIC-R/L.

Für *applname* ist der Anwendungsname der lokalen UTM-Anwendung anzugeben, über den die Verbindungen zwischen der UTM-Anwendung und den Client-Anwendungen aufgebaut werden.

Standard: Es wird der bei der KDCDEF-Generierung in MAX APPLINAME angegebene Anwendungsname angenommen.

STATUS= sperrt einen Client oder Drucker bzw. lässt gesperrte Clients und Drucker wieder zu.

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Cluster-global.

ON Die Clients/Drucker *ptermname_1,...,ptermname_10* werden wieder zugelassen.

OFF Die Clients/Drucker *ptermname_1,...,ptermname_10* sollen gesperrt werden.

Zulassung und Sperre wirken über das Beenden der Anwendung hinaus.

PRONAM=proname

Name des Prozessors, auf/an dem sich die Clients/Drucker *(ptermname_1,...,ptermname_10)* befinden. Hier ist der Prozessorname anzugeben, der beim Eintragen des Client/Druckers in die Konfiguration angegeben wurde.

Für Clients und Drucker einer UTM-Anwendung auf BS2000-Systemen ist die Angabe von *proname* Pflicht.

Bei einer UTM-Anwendung auf Unix-, Linux- und Windows-Systemen geben Sie *proname* nicht an, wenn die angegebenen Clients und Drucker lokal angeschlossen sind.

ACTION= legt fest, welche Aktion openUTM durchführen soll.

CON openUTM soll die logischen Verbindungen zu den Clients und Druckern *ptermname_1,..., ptermname_10* aufbauen.

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Knoten-lokal.

ACTION=CON ist **nicht** erlaubt für:

- UPIC-Clients
- Clients, die über einen LTERM-Pool mit der Anwendung verbunden werden
- Terminals, die über einen Multiplexanschluss mit der UTM-Anwendung auf BS2000-Systemen verbunden sind.
- Terminals einer UTM-Anwendung auf Unix-, Linux- und Windows-Systemen (PTYPE=TTY)

DIS openUTM soll die logischen Verbindungen zu den Clients und Druckern *ptermname_1,..., ptermname_10* abbauen.

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Knoten-lokal.

In openUTM auf BS2000-Systemen wird ACTION=DIS abgewiesen, wenn die Verbindung zu einem Terminal abgebaut werden soll, das über einen Multiplexanschluss mit der UTM-Anwendung verbunden ist und für das eine Session existiert, die sich im Zustand DISCONNECT-PENDING befindet. Der KDCPTERM-Aufruf wird abgewiesen. Die logische Verbindung muss mit ACTION=REL abgebaut werden.
Ob sich eine Session im Zustand DISCONNECT-PENDING befindet, können Sie mit KDCINF PTERM überprüfen.

REL (nur auf BS2000-Systemen)

ACTION=REL ist nur erlaubt, wenn die in *ptermname_1,...,ptermname_10* angegebenen Clients über einen Multiplexanschluss mit der Anwendung verbunden sind.

Mit ACTION=REL wird eine Session freigegeben. Die logische Verbindung zu dem Client wird abgebaut. Die Angabe von ACTION=REL ist nur zulässig, wenn sich die Session im Zustand DISCONNECT-PENDING mit abgelaufenem Timer (ca.10 Minuten) befindet.

ACON (automatic connection)
Bei den folgenden Starts der Anwendung soll openUTM automatisch logische Verbindungen zu *ptermname_1,...,ptermname_10* aufbauen.

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Cluster-global.

ACTION=ACON ist **nicht** erlaubt für:

- UPIC-Clients
- Clients, die über einen LTERM-Pool mit der Anwendung verbunden werden
- für Terminals einer UTM-Anwendung auf Unix-, Linux- und Windows-Systemen (PTYPE=TTY)

NACON (no automatic connection)
macht die Angabe ACON unwirksam, d.h. openUTM baut bei den folgenden Starts der Anwendung zu *ptermname_1,...,ptermname_10* keine logische Verbindung auf.

In UTM-Cluster-Anwendungen gilt: Der Operand wirkt Cluster-global.

Ausgabe von KDCPTERM

Am Administrator-Terminal werden die neuen und alten Eigenschaften (NEW, OLD) der angegebenen physischen Clients und Drucker angezeigt.

Die Ausgabe ist abhängig davon, ob einem PTERM ein kurzer oder ein langer Rechnername zugeordnet ist. Bei einem langen Rechnernamen wird die Information zu einem PTERM in zwei Bildschirmzeilen ausgegeben.

```
PTERM    PRONAM   BCAMAPPL  STATUS         CONNECTION
                            NEW    OLD     NEW         OLD
pterm1   proname  applname  ON|OFF ON|OFF  C|D|W A M   C|D|W A M    [POOL PTERM]
                                           T|E         T|E
pterm1   long.processor.name
                  applname  ON|OFF ON|OFF  C|D|W A M   C|D|W A M    [POOL PTERM]
                                           T|E         T|E
```

Erläuterungen zur Ausgabe

PTERM

Name des Client/Druckers

PRONAM

Name des Prozessors, auf/an dem sich der Client/Drucker befindet.

BCAMAPPL

Name der lokalen UTM-Anwendung, über den die Verbindungen zu dem Client/Drucker aufgebaut werden.

STATUS

Der Client/Drucker ist gesperrt (OFF) oder nicht gesperrt (ON)

CONNECTION

1. Spalte:

C/D/W	Client/Drucker ist z.Zt. mit der Anwendung verbunden (C) bzw. nicht verbunden (D) oder openUTM versucht gerade eine Verbindung aufzubauen (W; waiting for connection)
T/E	wird nur für Terminals ausgegeben, die über einen Multiplexanschluss mit einer UTM-Anwendung auf einem BS2000-System verbunden sind. T: (timer) Die Session ist im Zustand DISCONNECT-PENDING; der Timer für das Warten auf die Bestätigung des Verbindungsaufbaus läuft E: (expired) Die Session ist im Zustand DISCONNECT-PENDING und der Timer für das Warten auf die Bestätigung ist ohne Eintreffen der Bestätigung abgelaufen

2. Spalte:
A: automatischer Verbindungsaufbau beim Start der Anwendung

3. Spalte (nur bei UTM-Anwendungen auf BS2000-Systemen):
Der Client ist über einen Multiplexanschluss mit der Anwendung verbunden (M) oder nicht (keine Angabe).

POOL PTERM

wird ausgegeben, wenn der Client über LTERM-Pool angeschlossen ist.

12.15 KDCSEND - Nachricht an LTERM-Partner senden (BS2000- Systeme)

Mit KDCSEND können Sie Nachrichten an ein, mehrere oder alle aktiven Terminals einer UTM-Anwendung auf BS2000-Systemen senden. Als Nachricht sendet openUTM dann die Meldung K023 mit der angegebenen Nachricht als Insert. Sie wird standardmäßig in der Systemzeile am Terminal ausgegeben. Das Meldungsziel der Meldung K023 kann jedoch geändert werden. Ist für die UTM-Meldung K023 das Meldungsziel PARTNER ausgewählt, können Sie die Nachricht auch an eine, mehrere oder alle konnektierten TS-Anwendungen senden. Die Nachricht geht nur an Dialog-Partner (LTERM mit USAGE=D).

```
KDCSEND   MSG='message'
          [ ,LTERM={ ltermname | (ltermname_1,...,ltermname_10) | KDCALL } ]
```

Für die Administration über Message Queuing müssen Sie KDCSENDA angeben.

MSG='message'

Für *message* ist die zu sendende Nachricht anzugeben. Sie muss in Hochkommas eingeschlossen und darf nicht länger als 74 Zeichen sein.
Für ein Hochkomma im Meldungstext sind zwei Hochkommas zu schreiben.
Ist dem LTERM-Partner ein Terminal zugeordnet, dann wird die Nachricht in der Systemzeile angezeigt.

LTERM= gibt an, an welche LTERM-Partner die Meldung gesendet werden soll.

(ltermname_1,...,ltermname_10)

Namen der LTERM-Partner, an die die Nachricht gesendet werden soll. Sie können maximal 10 Namen angeben. Bei nur einem Namen können die Klammern entfallen.

KDCALL

Die Nachricht soll an alle aktiven LTERM-Partner gesendet werden, d.h. an alle Clients, zu denen derzeit eine logische Verbindung besteht.

Standard: KDCALL

Ausgabe von KDCSEND

Am Administrator-Terminal wird die Nachricht *message* angezeigt.

12.16 KDCSHUT - Anwendungslauf beenden

Mit KDCSHUT können Sie eine UTM-Anwendung beenden.

In UTM-Cluster-Anwendungen können Sie angeben, ob der Anwendungslauf auf allen Knoten beendet wird oder nur auf dem Knoten, an dem der Aufruf erfolgt.

Folgende Möglichkeiten stehen Ihnen zur Verfügung:

- Sie können den Anwendungslauf normal beenden. UTM beendet den Anwendungslauf, sobald alle laufenden Dialog-Schritte beendet sind (NORMAL).
- Sie können den Anwendungslauf zeitgesteuert nach einer angegebenen Zeitspanne beenden (WARN).
- Sie können die Anwendung beenden, nachdem alle UTM-D-Dialoge beendet und alle UTM-D-Verbindungen abgebaut sind, spätestens jedoch nach einer angegeben Zeitspanne (GRACE).
- Sie können die Anwendung abbrechen, d.h. sofort abnormal beenden (KILL).

Beachten Sie beim Anwendungsabbruch Folgendes:

Der Anwendungsabbruch kann nicht als Asynchron-Vorgang erfolgen, d.h. der Asynchron-Transaktionscode KDCSHUTA KILL hat keine Wirkung.

Beachten Sie beim Shutdown von Anwendungen mit verteilter Verarbeitung Folgendes:

- Beenden Sie Anwendungen mit verteilter Verarbeitung am besten mit KDCSHUT GRACE oder alternativ mit Warnung (KDCSHUT WARN).
  Die Verwendung von KDCSHUT GRACE oder WARN verringert die Wahrscheinlichkeit, dass Vorgänge abgebrochen werden und verteilte Transaktionen im Transaktionsstatus P (preliminary end of transaction) bleiben.
- Eine Anwendung mit verteilter Verarbeitung wird nicht normal beendet, wenn es zum Zeitpunkt des Shutdowns noch Vorgänge mit Transaktionsstatus P (perpare to commit) gibt, oder wenn für Asynchron-Nachrichten an einen Partner-Server noch keine Quittungen eingetroffen sind. openUTM gibt dann die Meldung K060 mit der Abbruchursache ENDPET aus. Es werden keine Dumps erzeugt.

  Daher sollten Sie bei KDCSHUT WARN oder GRACE eine Zeit angeben, die größer ist als die Zeit, die eine verteilte Transaktion maximal im Zustand PTC (d.h. Transaktionsstatus P) verbleibt. Damit verringert sich die Wahrscheinlichkeit, dass verteilte Transaktionen beim Anwendungsende noch in diesem Zustand sind und die Anwendung abnormal mit ENDPET beendet wird.

Zum Beenden einer UTM-Anwendung siehe auch das openUTM-Handbuch „Einsatz von UTM-Anwendungen“.

```
KDCSHUT       { GRACE [ ,TIME=time_min ] | KILL | NORMAL |
                WARN [ ,TIME=time_min ] }

              [ ,SCOPE= { LOCAL | GLOBAL } ]
```

Für die Administration über Message Queuing müssen Sie KDCSHUTA angeben.

GRACE
: Alle aktiven Verbindungen von LPAPs und OSI-LPAPs werden auf QUIET gesetzt. Die Anwendung wird beendet, sobald alle UTM-D-Verbindungen abgebaut sind, spätestens jedoch nach Ablauf der angegeben Zeit.

Auf BS2000-Sytemen wird an allen aktiven Terminals in der Systemzeile ein Hinweis auf den bevorstehenden Shutdown der Anwendung angezeigt, sowie ein Hinweis auf die bis zum Shutdown verbleibende Zeit (siehe TIME-Operand).

> ¡ Nach der Eingabe von KDCSHUT GRACE können sich nur noch Benutzer mit Administrationsberechtigung anmelden. Es können nur noch Vorgänge gestartet werden, deren Vorgangs-TAC zu einem Administrations-Teilprogramm gehört. Alle UTM-Benutzerkommandos außer KDCOUT werden noch ausgeführt.

KILL
Der Anwendungslauf wird abgebrochen, d.h. sofort beendet. Offene Vorgänge werden nicht mehr beendet. Von allen Prozessen wird ein UTM-Dump erstellt mit Dumpcode='ASIS99'.

NORMAL
Die Beendigung der Anwendung wird sofort eingeleitet. Es können sich keine Benutzer mehr bei der Anwendung anmelden, und Benutzer können keine neuen Vorgänge mehr beginnen. Neue Dialog-Eingaben werden nicht mehr bearbeitet. Ist die neue Dialog-Eingabe Teil einer Mehrschritt-Transaktion, dann wird die Mehrschritt-Transaktion auf den letzten Sicherungspunkt zurückgesetzt. Alle logischen Verbindungen zu Clients und Druckern werden abgebaut.
Offene Vorgänge können nach dem nächsten Start der Anwendung weiter bearbeitet werden.

WARN
Alle aktiven Verbindungen von LPAPs und OSI-LPAPs werden auf QUIET gesetzt.

Auf BS2000-Sytemen wird an allen aktiven Terminals in der Systemzeile ein Hinweis auf den bevorstehenden Shutdown der Anwendung angezeigt, sowie ein Hinweis auf die bis zum Shutdown verbleibende Zeit (siehe TIME-Operand).

> ¡ Nach der Eingabe von KDCSHUT WARN können sich nur noch Benutzer mit Administrationsberechtigung anmelden. Es können nur noch Vorgänge gestartet werden, deren Vorgangs-TAC zu einem Administrations-Teilprogramm gehört. Alle UTM-Benutzerkommandos außer KDCOUT werden noch ausgeführt.

TIME=time_min

wirkt nur zusammen mit WARN und GRACE.

Bedeutung bei GRACE:
time_min ist die Zeit in Minuten, nach deren Ablauf die Anwendung spätestens beendet wird.

Bedeutung bei WARN:
time_min ist die Zeit in Minuten, nach deren Ablauf die Anwendung beendet wird.

Maximal: 255 Minuten
Minimal: 1 Minute

Die Angabe TIME=0 wird von openUTM abgelehnt.

Hinweis für BS2000-Systeme:

- An allen aktiven Terminals wird in der Systemzeile ein Hinweis auf den bevorstehenden Shutdown der Anwendung angezeigt sowie ein Hinweis auf die bis zum Shutdown verbleibende

Zeit. Sind sehr viele Terminals aktiv (Konfigurationen mit vielen Terminals) benötigt das Ausgeben der Shutdown-Ankündigung eine gewisse Zeit. Deshalb sollten Sie TIME nicht zu klein wählen.

- Dem TAC KDCSHUT wurde bei der KDCDEF-Generierung die CPU-Zeit zugeordnet, die KDCSHUT maximal für die Ausführung des Shutdown verbrauchen darf. Diese Zeit sollte in Anwendungen mit vielen Terminals hinreichend groß gewählt sein. Reicht diese Zeit nicht aus, dann bricht openUTM den Vorgang ab und gibt die Meldung K017 aus.

SCOPE= gibt den Wirkungsbereich des Kommandos an. Dieser Parameter ist nur für UTM-Cluster-Anwendungen von Bedeutung.

LOCAL Das Kommando wirkt nur auf die lokale Knoten-Anwendung.Standardwert.

GLOBAL Das Kommando wirkt auf alle Knoten-Anwendungen der UTM-Cluster-Anwendung. SCOPE=GLOBAL wird abgelehnt, wenn nicht alle laufenden Knoten-Anwendungen gleich generiert sind. Dies kann z.B. der Fall sein, wenn eine Änderungsgenerierung der KDCFILE durchgeführt wird, ohne dass die UTM-Cluster-Anwendung vollständig beendet wurde.

Ausgabe von KDCSHUT

Am Administrator-Terminal wird die Meldung "COMMAND ACCEPTED" angezeigt.

Das tatsächliche Ende der Anwendung zeigt openUTM wie folgt an:

- BS2000-Systeme: Das Anwendungsende wird nur auf der Konsole angezeigt. Die Anzeige erfolgt, sobald sich der letzte Prozess der UTM-Anwendung beendet hat.
- Unix- und Linux-Systeme: Das Anwendungsende wird vom *utmmain*-Prozess nach *stdout* und *stderr* protokolliert.
- Windows-Systeme: Das Anwendungsende wird vom *utmmain*-Prozess nach *stdout* und *stderr* protokolliert. Wird eine Anwendung als Dienst/service gestartet, werden auch Meldungen in das Ereignisprotokoll (Eventlogging) des Windows-Systems geschrieben.

12.17 KDCSLOG - SYSLOG-Datei administrieren

Mit KDCSLOG können Sie die System-Protokolldatei SYSLOG im laufenden Betrieb administrieren. Sie können folgende Tätigkeiten ausführen:

- Die automatische Größenüberwachung der SYSLOG ein- und ausschalten.
- Den Schwellwert für die Größenüberwachung definieren bzw. ändern.
- Die SYSLOG-Datei auf die nächste Dateigeneration der SYSLOG-FGG umschalten.
- Den Inhalt des UTM-internen Meldungspuffers in die SYSLOG-Datei schreiben.
- Informationen über die Eigenschaften der SYSLOG-Datei anfordern.

Wirkung in UTM-Cluster-Anwendungen auf Unix-, Linux- und Windows-Systemen

Der Aufruf wirkt Cluster-global, d.h. die System-Protokolldatei SYSLOG wird für jede Knoten-Anwendung administriert. Die Größenüberwachung wirkt über den aktuellen Lauf der UTM-Cluster-Anwendung hinaus. Das Umschalten oder das Schreiben des Puffers wirkt nur für den aktuellen Anwendungslauf der UTM-Cluster-Anwendung, d.h. für alle Knoten-Anwendungen, die derzeit laufen.

Wirkungsdauer der Änderungen

Der zuletzt eingestellte Schwellwert für die Größenüberwachung wird auch nach dem erneuten Start der Anwendung eingestellt.

Liegt die Basis der SYSLOG-FGG innerhalb des gültigen Bereichs der SYSLOG-FGG (zwischen der ersten und der letzten Dateigeneration), dann protokolliert openUTM im nächsten Anwendungslauf zunächst in die Basisdateigeneration. Liegt die Basis außerhalb des gültigen Bereichs, dann legt openUTM für das Protokoll eine neue Dateigeneration an.

```
KDCSLOG     { INFO | WRITE | SIZE=fg_size | SWITCH [ , SIZE=fg_size ] }
```

Für die Administration über Message Queuing müssen Sie KDCSLOGA angeben.

INFO
: Es soll Information über die SYSLOG-Datei bzw. SYSLOG-FGG angezeigt werden. Die Beschreibung der Ausgabe finden Sie im Anschluss an die Operandenbeschreibung.

WRITE
: Alle an das Meldungsziel SYSLOG ausgegebenen Meldungen, die noch im UTM-internen Meldungspuffer zwischengespeichert sind, werden sofort in die aktuelle SYSLOG-Datei geschrieben.

  Diese Funktion ist nützlich, wenn die SYSLOG-Datei, die als einfache Datei angelegt wurde, im laufenden Betrieb ausgewertet werden soll. Es werden dann alle Meldungen mit dem Meldungsziel SYSLOG in der Auswertung berücksichtigt, die bis zu diesem Zeitpunkt von openUTM erzeugt wurden.

  Für die Auswertung der SYSLOG im laufenden Betrieb ist es jedoch besser, die SYSLOG als FGG anzulegen. Dann können Sie vor Beginn der Auswertung die Dateigeneration mit KDCSLOG SWITCH umschalten und erfassen somit für die Auswertung alle bis zum Umschaltzeitpunkt von openUTM erzeugten Meldungen. D.h. openUTM schreibt den Meldungspuffer automatisch vor dem Umschalten in die „alte“ SYSLOG-Datei. Es gelangen jedoch keine Meldungen in die Auswertung, die nach dem Umschalt-Zeitpunkt erzeugt werden.

SIZE=fg_size

Das Kommando KDCSLOG SIZE=*fg_size* wird nur ausgeführt, wenn die SYSLOG als FGG angelegt ist.

fg_size legt den Schwellwert für die automatische Größenüberwachung der SYSLOG-Datei neu fest. Für *fg_size* ist der gewünschte Schwellwert als Anzahl an UTM-Seiten anzugeben (z.B. SIZE=100 definiert einen Schwellwert von 100 * Größe einer UTM-Seite).

Es muss *fg_size* >= 0 angegeben werden. Wird *fg_size* < 0 angegeben, lehnt openUTM die Kommandoausführung ab.
Mit fg_size=0 können Sie die automatische Größenüberwachung ausschalten. Durch *fg_size* > 0 wird die automatische Größenüberwachung eingeschaltet. Angaben für *fg_size*, die zwischen 1 und 99 liegen, werden automatisch durch 100 ersetzt. Werte >= 100 werden unverändert als Schwellwert übernommen.

Minimalwert: 100
Maximalwert: (2^{31} -1)

SWITCH

wird nur ausgeführt, wenn die SYSLOG als FGG angelegt ist.
KDCSLOG SWITCH veranlasst openUTM, die SYSLOG-Datei auf die nächste Dateigeneration weiterzuschalten.

openUTM garantiert, dass nach dem erfolgreichen Ausführen dieses Kommandos keine Meldungen mehr in die alte SYSLOG-Dateigeneration geschrieben werden.

Vor dem Umschalten auf eine neue Dateigeneration schreibt openUTM noch die im internen Meldungspuffer zwischengespeicherten Meldungen in die alte Dateigeneration.

In UTM-Anwendungen auf BS2000-Systemen ist Folgendes zu beachten:

- Eine erfolgreiche Kommandobearbeitung von KDCSLOG SWITCH durch openUTM heißt nicht, dass Sie sofort über die alte Dateigeneration verfügen können. Die alte Dateigeneration wird eventuell noch längere Zeit von UTM-Prozessen offengehalten, z.B. weil die Bearbeitung eines Teilprogramms, das vor dem Umschalten gestartet wurde, noch nicht abgeschlossen ist und noch keine Meldung mit Meldungsziel SYSLOG von dem zugehörigen Prozess geschrieben wurde.
- Sie können mit KDCSLOG INFO abfragen, welche SYSLOG-Dateigenerationen bereits von allen UTM-Prozessen geschlossen worden sind. Das sind alle Dateigenerationen kleiner LOWEST-OPEN-GEN (siehe Beschreibung der Ausgabe im Abschnitt "KDCSLOG - SYSLOG-Datei administrieren").

SWITCH,SIZE=fg_size

wird nur ausgeführt, wenn die SYSLOG als FGG angelegt ist.

Mit KDCSLOG SWITCH,SIZE=*fg_size* können Sie die SYSLOG auf eine neue Dateigeneration umschalten und gleichzeitig den Schwellwert für die automatische Größenüberwachung der folgenden Dateigenerationen neu festlegen. Dabei garantiert openUTM, dass entweder beide Aktionen erfolgreich ausgeführt werden oder keine. D.h. nur wenn das Umschalten der SYSLOG erfolgreich war, stellt openUTM den neuen Schwellwert ein.

Kann openUTM nicht auf die folgende Dateigeneration umschalten, dann wird der Schwellwert nicht geändert. Die Größenüberwachung wird suspendiert und openUTM ignoriert den für *fg_size* angegebenen Wert. Erst durch einen folgenden erfolgreichen Umschaltversuch (KDCSLOG

SWITCH) kann die Größenüberwachung wieder eingestellt werden. Wurde dabei *fg_size* nicht angegeben, übernimmt openUTM den „alten“ Wert von *fg_size* als Schwellwert.

Die Funktion, die Einschränkungen und die möglichen Werte von *fg_size* finden Sie bei der Beschreibung zu den Operanden SWITCH und SIZE=*fg_size*.

Ausgaben von KDCSLOG INFO

```
SYSLOG FILE NAME               filename
FILE GENERATION GROUP          fgg
LAST SWITCH                    last-switch
SIZE CONTROL                   control
CURRENT SYSLOG SIZE            csp UTM PAGE(S)   =  csk KB
SIZE CONTROL VALUE             scp UTM PAGE(S)   =  sck KB
SYSLOG FILE                    rel% FILLED
FILE GENERATIONS OF APPL       START-GEN           start-gen
                               LOWEST-OPEN-GEN     low-gen
                               CURRENT-GEN         curr-gen
FILE GENERATIONS               BASE-GEN            basis-gen
                               FIRST-GEN           first-gen
                               LAST-GEN            last-gen
```

Erläuterungen zur Ausgabe

SYSLOG FILE NAME

Name der aktuellen SYSLOG-Datei. Ist die SYSLOG als FGG angelegt, so wird die Generationsnummer der aktuellen Dateigeneration mit angezeigt.

FILE GENERATION GROUP

zeigt an, ob die SYSLOG als FGG oder als einfache Datei angelegt ist.

YES
Die SYSLOG ist als FGG angelegt.

NO
Die SYSLOG ist als einfache Datei angelegt.

LAST SWITCH

wird nur ausgegeben, wenn die SYSLOG als FGG angelegt ist.
LAST SWITCH gibt an, ob der letzte Versuch von openUTM, auf die nächste Dateigeneration umzuschalten, fehlerfrei abgelaufen ist. Folgende Werte sind möglich:

SUCCESSFUL
Der letzte Umschaltversuch ist fehlerfrei abgelaufen.

FAILED
Beim letzten Umschaltversuch von openUTM ist ein Fehler aufgetreten. openUTM konnte nicht auf die nächste Dateigeneration umschalten.

NONE
Es gab im aktuellen Anwendungslauf noch keinen Umschaltversuch.

SIZE CONTROL

wird nur ausgegeben, wenn die SYSLOG als FGG angelegt ist.
SIZE CONTROL gibt an, ob die automatische Größenüberwachung eingeschaltet ist. Folgende Werte sind möglich:

ON
Größenüberwachung ist eingeschaltet

OFF
Größenüberwachung ist ausgeschaltet

SUSPENDED
Der letzte Versuch auf eine andere Dateigeneration umzuschalten ist fehlgeschlagen (bei LAST SWITCH wird FAILED ausgegeben). Aus diesem Grund ist die Größenüberwachung suspendiert.
Maßnahme: Sie können mit KDCSLOG SWITCH erneut versuchen, die SYSLOG umzuschalten. Verläuft das Umschalten fehlerfrei, so wird die Größenüberwachung durch openUTM automatisch wieder aktiviert.

CURRENT SYSLOG SIZE

Momentane Größe der SYSLOG-Datei/aktuellen Dateigeneration; ausgegeben in Anzahl UTM-Seiten (*csp*) und in KB (*csk*).

Alle folgenden Informationen werden nur ausgegeben, wenn die SYSLOG als FGG angelegt ist.

SIZE CONTROL VALUE

Eingestellter Größenschwellwert der automatischen Größenüberwachung. Ausgegeben wird der Schwellwert in Anzahl UTM-Seiten (*scp*) und in KB (*sck*). Bei sehr großen Schwellwerten wird der Kilobyte-Wert nicht angezeigt (z.B. bei 2^{31} KB).
Wird bei SIZE CONTROL VALUE 0 ausgegeben, dann ist die Größenüberwachung ausgeschaltet.

SYSLOG FILE .... % FILLED

wird ausgegeben, falls die automatische Größenüberwachung eingeschaltet ist. Der Wert gibt den Füllgrad der SYSLOG-Datei relativ zum eingestellten Größenschwellwert (SIZE CONTROL VALUE) in Prozent an. Ist die Größenüberwachung durch openUTM suspendiert, dann kann der Füllgrad der SYSLOG-Datei auch größer als 100% sein. In diesem Fall wird für SYSLOG FILE „>100% FILLED“ ausgegeben.

START-GEN Generationsnummer der ersten SYSLOG-Dateigeneration, die openUTM im aktuellen Anwendungslauf beschrieben hat.

LOWEST-OPEN-GEN

Generationsnummer der ältesten SYSLOG-Dateigeneration, die noch von einem Prozess der Anwendung offengehalten wird.

CURRENT-GEN

Generationsnummer der Dateigeneration, in die openUTM gerade protokolliert.

BASE-GEN Generationsnummer der eingestellten Basis der SYSLOG-FGG.

FIRST-GEN Generationsnummer der ersten gültigen Dateigeneration der SYSLOG-FGG.

Auf BS2000-Systemen entspricht das dem FIRST-GEN aus dem SHOW-FILE-ATTRIBUTES-Kommando.

LAST-GEN Generationsnummer der letzten gültigen Dateigeneration der SYSLOG-FGG.

Auf BS2000-Systemen entspricht das dem LAST-GEN aus dem SHOW-FILE-ATTRIBUTES-Kommando.

Ausgabe von KDCSLOG WRITE

1. Kann openUTM den Meldungspuffer ordnungsgemäß in die SYSLOG schreiben, dann gibt openUTM folgende Meldung aus:

   ```
   **** SYSLOG BUFFER WRITTEN ****
   ```

2. Ist der Meldungspuffer bei der Kommandobearbeitung leer, wird Folgendes ausgegeben:

   ```
   **** SYSLOG BUFFER IS EMPTY ****
   ```

3. Kann openUTM den Meldungspuffer nicht ordnungsgemäß in die SYSLOG schreiben, dann wird folgende Meldung ausgegeben:

   ```
   **** SYSLOG BUFFER NOT WRITTEN ****
   ```

Ausgabe von KDCSLOG SIZE=fg_size

1. Wurde *fg_size* >= 0 angegeben und ist die SYSLOG der Anwendung als FGG angelegt, dann wird z.B. folgender Text ausgegeben:

   ```
          NEW      OLD
    SIZE      100      0  COMMAND ACCEPTED - MINIMAL SIZE TAKEN
   ```

   Der Zusatztext COMMAND ACCEPTED- MINIMAL SIZE TAKEN wird nur ausgegeben, wenn für *fg_size* ein Wert zwischen 1 und 99 eingegeben wurde.

2. Ist die SYSLOG nicht als FGG angelegt, wird folgende Meldung ausgegeben:

   ```
   COMMAND REJECTED - SYSLOG FILE IS NO FGG
   ```

Ausgabe von KDCSLOG SWITCH

1. Konnte openUTM die SYSLOG erfolgreich umschalten, dann wird folgende Meldung ausgegeben:

   ```
   *** SYSLOG SWITCH ACCEPTED ***
   ```

2. Ist die SYSLOG nicht als FGG angelegt, so wird folgende Meldung ausgegeben:

   ```
   *** SYSLOG SWITCH REJECTED - SYSLOG FILE IS NO FGG ***
   ```

3. Tritt beim Umschalten ein Fehler auf, so gibt openUTM folgende Meldung aus:

   ```
   *** SYSLOG SWITCH REJECTED ***
   ```

12.18 KDCSWTCH - Zuordnung Clients, Drucker zu LTERM- Partnern ändern

Mit KDCSWTCH können Sie die Zuordnung von Clients und Druckern (PTERM) zu LTERM-Partnern neu festlegen.

KDCSWTCH ist nur in stand-alone UTM-Anwendungen erlaubt.

KDCSWTCH hat folgende Wirkung:

- die bestehende Zuordnung des Client/Druckers zu einem LTERM-Partner wird aufgelöst und
- der Client/Drucker wird dem angegebenen LTERM-Partner zugeordnet.

Die Funktion kann nur ausgeführt werden, wenn keine logische Verbindung zwischen dem Client/Drucker und der UTM-Anwendung besteht.

Mit KDCSWTCH können Sie z.B. einem Druckerbündel einen weiteren Drucker zuordnen. Bei einem Druckerbündel sind mehrere physische Drucker einem LTERM-Partner zugeordnet.

Wollen Sie einem Drucker einen LTERM-Partner zuordnen, dem wiederum ein Druckersteuer-LTERM zugeordnet ist (CTERM), dann muss die Control Identification des Druckers (CID) im Bereich des Druckersteuer-LTERMs eindeutig sein.

> openUTM auf Windows-Systemen unterstützt keine Drucker.

Einschränkung

Eine neue Zuordnung des LTERM-Partners ist nur für Terminals und Drucker möglich. Die bei der Konfiguration festgelegte Zuordnung zu einem LTERM-Partner kann nicht geändert werden

- bei UPIC-Clients
- bei TS-Anwendungen (APPLI/SOCKET), die als Dialog-Partner generiert sind
- bei Clients, die sich über einen LTERM-Pool an die Anwendung anbinden
- bei LTERMs, die zu einem LTERM-Bündel oder zu einer LTERM-Gruppe gehören

Wenn Sie einem Terminal oder einem Drucker einen neuen LTERM-Partner zuweisen, dann darf der LTERM-Partner keinem Client/Drucker eines anderen Protokoll-Typs zugewiesen sein (weder aktuell noch in der Vergangenheit). Dabei sind hier die vier Protokoll-Typen Terminals, TS-Anwendungen, Drucker und RSO-Drucker zu unterscheiden.

KDCSWTCH wird also abgewiesen, wenn:

- der in *ptermname* angegebene Client ein UPIC-Client oder eine TS-Dialog-Anwendung ist (PTYPE=UPIC-R/L oder PTYPE=APPLI/SOCKET) oder
- der in *ltermname* angegebene LTERM-Partner bisher einem UPIC-Client oder einer TS-Dialog-Anwendung zugeordnet ist oder
- der in *ltermname* angegebene LTERM-Partner einem LTERM-Pool zugeordnet ist, oder
- auf BS2000-Systemen der in *ptermname* angegebene Drucker ein RSO-Drucker ist (PTYPE=RSO) und der in *ltermname* angegebene LTERM- Partner bisher einem normalen Drucker zugeordnet war
- der in *ltermname* angegebene LTERM-Partner zu einer LTERM-Gruppe oder einem LTERM-Bündel gehört.

Wirkungsdauer der Änderungen

Die Änderungen wirken über das Ende der Anwendung hinaus.

BS2000-Systeme:

```
KDCSWTCH ltermname,ptermname,proname [ ,applname ]
```

Unix-, Linux- und Windows-Systeme:

```
KDCSWTCH ltermname,ptermname [,proname [ ,applname ] ]
```

Für die Administration über Message Queuing müssen Sie KDCSWCHA angeben.

ltermname Name des LTERM-Partners, dem der Client oder Drucker zugeordnet werden soll. Der LTERM-Partner muss in der Konfiguration der UTM-Anwendung existieren.

ptermname, proname, applname

identifizieren den Client/Drucker eindeutig.

ptermname

Name des Client oder Druckers (PTERM-Name)

proname Name des Prozessors, auf dem der Client abläuft bzw. an dem der Client oder Drucker angeschlossen ist.

Die Angabe von *proname* ist in UTM-Anwendungen auf BS2000-Systemen Pflicht.

In UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen muss *proname* nur angegeben werden, wenn der Client oder Drucker nicht lokal angeschlossen ist.

Standardwert in openUTM auf Unix-, Linux- und Windows-Systemen: Leerzeichen für lokale Clients/Drucker.

applname Die Angabe ist nur bei UPIC-Clients und TS-Anwendungen sinnvoll.
Für *applname* geben Sie den Namen der UTM-Anwendung an, der dem Client beim Eintragen in die Konfiguration zugeordnet wurde.

Die Angabe von *applname* ist Pflicht, wenn der dem Client zugeordnete BCAMAPPL-Name nicht mit dem Namen der UTM-Anwendung übereinstimmt, der bei der KDCDEF-Generierung in MAX APPLINAME festgelegt wurde. Wird in diesem Fall *applname* nicht angegeben, wird das Kommando abgewiesen mit der Meldung:

```
BCAMAPPL-NAME 'applname' INVALID OR NOT DEFINED
```

Standard:
Name der Anwendung, der in der KDCDEF-Steueranweisung MAX im Operanden APPLINAME definiert wurde.

Ausgabe von KDCSWTCH

Am Administrator-Terminal wird die neue und alte Zuordnung zwischen Client/Drucker und LTERM-Partner angezeigt.

Die Ausgabe ist abhängig davon, ob einem PTERM ein kurzer oder ein langer Rechnername zugeordnet ist. Bei einem langen Rechnernamen wird die Information zu einem PTERM in zwei Bildschirmzeilen ausgegeben.

Im Folgenden finden Sie die Ausgaben für die Aufrufe mit einem kurzen und mit einem langen Rechnernamen.

Hier soll dem Client *pterm1* der LTERM-Partner *lterm2* zugeordnet werden.

```
KDCSWTCH lterm2,pterm1,proname1,app11
PTERM    | PRONAM   | BCAMAPPL | NEW LTERM || OLD LTERM
---------+----------+----------+-----------++----------
pterm1   | proname1 | appl1    | lterm2    || lterm1
---------+----------+----------+-----------++----------
pterm2   | proname2 | appl2    |           || lterm2
```

```
KDCSWTCH lterm2,pterm1,long.processor.name1,appl1
PTERM    | PRONAM   | BCAMAPPL | NEW LTERM || OLD LTERM
---------+----------+----------+-----------++----------
pterm1   | long.processor.name1
                    | appl1    | lterm2    || lterm1
---------+----------+----------+-----------++----------
pterm2   | proname2 | appl2    |           || lterm2
```

Erläuterungen zur Ausgabe

openUTM gibt die alte und neue Zuordnung für den Client aus, der beim Aufruf von KDCSWTCH in *ptermname* angegeben wurde (hier *pterm1*), und für den Client, der dem LTERM-Partner *lterm2* vor dem KDCSWTCH-Aufruf zugeordnet war (hier *pterm2*).

Vor dem Aufruf von KDCSWTCH war dem Client *pterm1* der LTERM-Partner *lterm1* zugeordnet, dem LTERM-Partner *lterm2* der Client *pterm2* (siehe Spalte OLD LTERM). Beide Zuordnungen werden durch den KDCSWTCH-Aufruf aufgehoben.

> **i** Sind *pterm1* und *pterm2* Clients (keine Drucker), dann werden die alten Zuordnungen von LTERM-Partner zu PTERM gelöscht.
> Sind *pterm1* und *pterm2* Drucker, dann wird die alte Zuordnung von *lterm2* zu *pterm2* nicht gelöscht. Es wird dann immer ein Druckerbündel erzeugt, d.h. dann sind beide Drucker dem LTERM-Partner *lterm2* zugeordnet.

PTERM — Name des Client oder Druckers.

PRONAM — Name des Prozessors, auf/an dem sich der Client/Drucker befindet.

BCAMAPPL — Name der lokalen UTM-Anwendung, über den die Verbindung zum Client/Drucker aufgebaut wird.

NEW LTERM — Name des LTERM-Partners, dem der Client/Drucker durch den KDCSWTCH-Aufruf zugeordnet wurde.

ltermname1

Name des LTERM-Partners, der dem Client/Drucker mit KDCSWTCH zugeordnet wurde.

ltermname2

Name des LTERM-Partners, der dem Client/Drucker vor dem KDCSWTCH zugeordnet war.

OLD LTERM Name des LTERM-Partners, dem der Client/Drucker bisher zugeordnet war.

Beispiel: Zusammenschalten von Druckern zu Druckerbündeln

Die Drucker *pterm1* und *pterm2* sollen zu einem Druckerbündel zusammengefasst werden. Der LTERM-Partner des Druckerbündels soll *lt-bundle* sein.

Zuordnung vor dem KDCSWTCH:
Der Drucker *pterm2* ist bereits dem LTERM-Partner *lt-bundle* zugeordnet. Der Drucker *pterm1* ist dem LTERM-Partner *lt-print* zugeordnet.

Aufruf:

```
KDCSWTCH lt-bundle,pterm1,proname1,appl1
```

Ausgabe:

```
PTERM    | PRONAM   | BCAMAPPL | NEW LTERM || OLD LTERM
---------+----------+----------+-----------++----------
pterm1   | proname1 | appl1    | lt-bundle || lt-print
---------+----------+----------+-----------++----------
pterm2   | proname2 | appl2    | lt-bundle || lt-bundle
```

12.19 KDCTAC - Transaktionscodes und TAC-Queues sperren, wieder freigeben

Mit KDCTAC können Sie Transaktionscodes und TAC-Queues sperren und Sperren aufheben, die bei der Generierung oder durch die Administration gesetzt wurden.
Außer auf den Transaktionscode KDCTAC ist die Funktion auf alle Transaktionscodes und TAC-Queues der Anwendung anwendbar.

Wirkung in UTM-Cluster-Anwendungen

KDCTAC wirkt in UTM-Cluster-Anwendungen Cluster-global.

Wirkungsdauer der Änderungen

Das Sperren des ereignisgesteuerten Vorgangs KDCMSGTC wirkt höchstens für den aktuellen Anwendungslauf. Für alle anderen TACs bleibt die Änderung über das Anwendungsende hinaus erhalten.

```
KDCTAC     TAC={ tacname | (tacname_1,tacname_2,...,tacname_10) }
           ,STATUS={ OFF | HALT | KEEP | ON }
```

Für die Administration über Message Queuing müssen Sie KDCTACA angeben.

TAC=(tacname_1,...,tacname_10)

Namen der zu administrierenden Transaktionscodes oder TAC-Queues. Sie können pro Aufruf maximal 10 Transaktionscodes bzw. TAC-Queues angeben. Bei nur einem TAC-Namen können die Klammern entfallen.

In der Liste darf der Transaktionscode KDCTAC nicht enthalten sein.

STATUS=

OFF die Transaktionscodes bzw. TAC-Queues *tacname_1,...,tacname_10* sollen gesperrt werden.

Transaktionscodes:

Mit STATUS=OFF können Sie nur Vorgangs-TACs sperren, d.h. TACs, die mit CALL=FIRST oder CALL=BOTH konfiguriert sind. Eine Sperre mit OFF bewirkt, dass openUTM ab sofort keine Aufträge mehr für diesen TAC annimmt. Wird ein TAC gesperrt, der mit CALL=BOTH konfiguriert ist, dann kann er trotzdem noch als Folge-TAC in einem Vorgang aufgerufen werden.

TAC-Queues:

Die TAC-Queues werden für Schreibzugriffe gesperrt; Lesezugriffe sind möglich.

HALT die Transaktionscodes bzw. TAC-Queues *tacname_1,...,tacname_10* sollen vollständig gesperrt werden.

Transaktionscodes:

Die vollständige Sperre eines TACs bewirkt, dass ab sofort keine Teilprogrammläufe mehr für diesen TAC gestartet werden. D.h. für den TAC werden keine Aufträge mehr angenommen und darüber hinaus ist er auch als Folge-TAC in einem Asynchron- oder Dialog-Vorgang gesperrt.

Wird ein vollständig gesperrter TAC als Folge-TAC aufgerufen, dann wird der Vorgang mit PEND ER (74Z) beendet. Asynchronaufträge, die bereits in die Message Queue des TACs eingereiht sind, werden nicht gestartet. Sie bleiben in der Message Queue, bis der TAC wieder freigegeben (STATUS=ON) oder auf STATUS=OFF gesetzt wird.

TAC-Queues:

Die TAC-Queues werden für Schreib- und Lesezugriffe gesperrt.

KEEP darf nur für TAC-Queues und Asynchron-Transaktionscodes angegeben werden, die auch Vorgangs-TACs (CALL=FIRST/BOTH) sind.

Transaktionscodes:

Der Transaktionscode wird gesperrt. Aufträge für den Transaktionscode werden zwar angenommen, jedoch nicht bearbeitet. Die Aufträge werden lediglich in die Auftragswarteschlange des Transaktionscodes geschrieben. Sie werden erst bearbeitet, wenn Sie den Status des Transaktionscodes ändern in ON.

Den Status KEEP können Sie benutzen, um Aufträge zu sammeln, die erst zu einem Zeitpunkt ausgeführt werden sollen, an dem die Anwendung weniger belastet ist (z.B. nachts).

Um eine Überlastung des Pagepools durch zuviele zwischengespeicherte Aufträge zu vermeiden, sollten Sie die Auftragswarteschlange des Transaktionscodes begrenzen. Dazu müssen Sie beim Erzeugen des Transaktionscodes den Parameter QLEV entsprechend setzen.

TAC-Queues:

Die TAC-Queues werden für Lesezugriffe gesperrt; Schreiben ist weiterhin möglich.

ON Die Transaktionscodes bzw. TAC-Queues *tacname_1,...,tacname_10* werden freigegeben. Eine von der Generierung oder durch die Administration gesetzte Sperre wird aufgehoben.

Ausgabe von KDCTAC

Am Administrator-Terminal werden die neuen und alten Eigenschaften der Transaktionscodes angezeigt.

TAC	STATUS	
	NEW	OLD
tacname_1	ON|OFF|HLT|KP	ON|OFF|HLT|KP
tacname_2	ON|OFF|HLT|KP	ON|OFF|HLT|KP

12.20 KDCTCL - Prozess-Anzahl einer TAC-Klasse ändern

> ¡ Der Aufruf des Kommandos KDCTCL ist nur sinnvoll, wenn in Ihrer Anwendung die Bearbeitung von Aufträgen mit dem Verfahren „Beschränkung der Prozesszahl für TAC-Klassen“ gesteuert wird, d.h. keine TAC-PRIORITIES-Anweisung generiert ist (siehe openUTM-Handbuch „Anwendungen generieren“).

Mit KDCTCL können Sie:

- sich über die aktuellen Werte, die für die TAC-Klasse eingestellt sind, informieren. Dazu geben Sie KDCTCL ohne die Operanden TASKS und TASKSFREE an.
- die Anzahl der Prozesse, die maximal gleichzeitig TACs einer TAC-Klasse bearbeiten dürfen, verändern. Eine Änderung ist nur erlaubt, wenn die KDCDEF-Generierung Ihrer Anwendung keine TAC-PRIORITIES-Anweisung enthält.

  Die Anzahl der Prozesse, die Sie für einzelne TAC-Klassen zulassen können, ist begrenzt durch die maximalen Prozesszahlen, die bei der KDCDEF-Generierung in der MAX-Anweisung festgelegt wurden (Operanden TASKS, ASYNTASKS und TASKS-IN-PGWT).
  Wenn Sie höhere Prozesszahlen eingeben, wird KDCTCL abgewiesen.

  Die nach dem KDCTCL tatsächlich eingestellte Anzahl der Prozesse, die gleichzeitig TACs einer TAC-Klasse bearbeiten, kann kleiner sein als der durch KDCTCL eingestellte Wert. Die tatsächliche Prozesszahl ist abhängig von der aktuellen Anzahl der Prozesse für die gesamte Anwendung (eingestellt über den Startparameter TASKS oder durch die Administration z.B. mit KDCAPPL).

Die maximale Prozessanzahl für eine TAC-Klasse können Sie auf zwei Arten festlegen; entweder geben Sie an, wieviele Prozesse gleichzeitig TACs der TAC-Klasse bearbeiten dürfen (Operand TASKS), oder Sie geben an, wieviele Prozesse von den Prozessen der Anwendung mindestens für die Bearbeitung der TACs anderer TAC-Klassen freigehalten werden sollen (Operand TASKSFREE). Der Unterschied zwischen TASKS und TASKSFREE ist folgender:

- Bei Verwendung von TASKS ist die maximale Anzahl der Prozesse, die für die angegebene TAC-Klasse zur Verfügung stehen, unabhängig von der Anzahl der Prozesse, die aktuell für das gesamte Anwendungsprogramm zur Verfügung stehen. D.h. die Prozesszahl der TAC-Klasse bleibt konstant, auch wenn die Prozesszahl der gesamten Anwendung herabgesetzt wird. Das gilt solange, bis die Prozesszahl der TAC-Klasse und die Prozesszahl des gesamten Anwendungsprogramms gleich groß sind.

  Durch die Verwendung des Operanden TASKS können (im Extremfall) Prozesse einer TAC-Klasse alle anderen TAC-Klassen behindern.
- Bei Verwendung von TASKSFREE ist die maximale Anzahl der Prozesse, die für die angegebene TAC-Klasse zur Verfügung stehen, dynamisch von der Anzahl der Prozesse abhängig, die aktuell für das gesamte Anwendungsprogramm zur Verfügung stehen. Für Prozesse anderer TAC-Klassen wird immer die in TASKSFREE angegebene Reserve freigehalten.

  Die maximale Anzahl von Prozessen für eine TAC-Klasse ergibt sich dann wie folgt:

  - Dialog-TAC-Klassen (1- 8): aktuelle Anzahl aller Prozesse, die für Dialog-TACs des gesamten Anwendungsprogramms zur Verfügung stehen (TASKS), abzüglich TASKSFREE, aber mindestens ein Prozess
  - Asynchron-TAC-Klassen (9-16): aktuelle Anzahl aller Prozesse, die für Asynchron-TACs des gesamten Anwendungsprogramms zur Verfügung stehen (TASKS), abzüglich TASKSFREE.

Wirkung in UTM-Cluster-Anwendungen

KDCTCL wirkt in UTM-Cluster-Anwendungen Knoten-lokal.

Wirkungsdauer der Änderung

Die Änderung wirkt nicht über das Anwendungsende hinaus. Die verfügbare Prozess-Anzahl wird immer durch den zuletzt eingegebenen KDCTCL-Aufruf bestimmt.

```
KDCTCL     CLASS=tacclass
           [ ,{ TASKS=number_tasks | TASKSFREE=number_tasks } ]
```

Für die Administration über Message Queuing müssen Sie KDCTCLA angeben.

CLASS=tacclass

Nummer der TAC-Klasse, für die die Anzahl der Prozesse geändert werden soll. Für *tacclass* können Sie eine Zahl zwischen 1 und 16 angeben (1 <= *tacclass* <= 16).

TASKS=number_tasks

darf nur angegeben werden, wenn für die Anwendung keine Prioritätensteuerung generiert ist, d.h. die Anwendung ohne TAC-PRIORITIES generiert ist.

In TASKS geben Sie an, wieviele Prozesse der Anwendung gleichzeitig TACs der TAC-Klasse *tacclass* bearbeiten dürfen.
Mit TASKS legen Sie eine absolute Anzahl von Prozessen für eine TAC-Klasse fest.

Minimalwert von *number_tasks*:
Für Dialog-TACs (Klasse 1-8) muss *number_tasks* >= 1 sein, da sonst Dialog-Vorgänge blockiert würden und die Benutzer am Terminal warten müssten, bis wieder Prozesse zugelassen werden.

Für Asynchron-TACs (Klasse 9-16) darf *number_tasks*=0 sein.

Maximalwert von *number_tasks*:
Der erlaubte Maximalwert für *number_tasks* ist abhängig von folgenden Faktoren:

- davon, ob die TAC-Klasse mit PGWT=YES generiert ist oder mit PGWT=NO.
  PGWT=YES bedeutet, dass in der TAC-Klasse Teilprogramme mit blockierenden Aufrufen (z.B. KDCS-Aufruf PGWT) ablaufen können.
- von den in der KDCDEF-Steueranweisung MAX statisch generierten Werten für TASKS, TASKS-IN-PGWT und ASYNTASKS.

Die erlaubten Wertebereiche für TASKS entnehmen Sie bitte der folgenden Tabelle.

TAC-Klasse	PGWT=	erlaubter Maximalwert
1 - 8 (Dialog-TACs)	NO	TASKS *)
	YES	TASKS-IN-PGWT *)
9 - 16 (Asynchron-TACs)	NO	ASYNTASKS *)
	YES	der kleinere der Werte: ASYNTASKS und *) TASKS-IN-PGWT *)

*) wie in der KDCDEF-Steueranweisung MAX statisch generiert

TASKSFREE=number_tasks

darf nur angegeben werden, wenn für die Anwendung keine Prioritätensteuerung generiert ist, d.h. die Anwendung ohne TAC-PRIORITIES generiert ist.

In TASKSFREE legen Sie fest, wieviele Prozesse der Anwendung für die Verarbeitung anderer TAC-Klassen als der angegebenen freigehalten werden sollen.

Ist *number_tasks* größer als die Zahl der Prozesse, die aktuell für das gesamte Anwendungsprogramm zur Verfügung stehen, dann passiert Folgendes:

- ist *tacclass* eine Dialog-TAC-Klasse, dann steht dieser weiterhin ein Prozess zur Bearbeitung ihrer TACs zur Verfügung
- ist *tacclass* eine Asynchron-TAC-Klasse, dann ist die Anzahl der ihr zur Verfügung stehenden Prozesse=0

Minimalwert von *number_tasks*: 0

Maximalwert von *number_tasks*:
Der erlaubte Maximalwert für *number_tasks* ist abhängig von den in der KDCDEF-Steueranweisung MAX statisch generierten Werten für TASKS und ASYNTASKS.

Die erlaubten Wertebereiche für TASKSFREE entnehmen Sie bitte der folgenden Tabelle.

TAC-Klasse	PGWT=	erlaubter Maximalwert
1 - 8 (Dialog-TACs)	NO	TASKS - 1 *)
	YES	TASKS - 1 *)
9 - 16 (Asynchron-TACs)	NO	ASYNTASKS *)
	YES	ASYNTASKS *)

*) wie in der KDCDEF-Steueranweisung MAX statisch generiert

Ausgabe von KDCTCL

Wenn Sie KDCTCL ohne TASKS bzw. TASKSFREE eingeben, werden Ihnen nur die aktuell eingestellten Werte angezeigt. Ansonsten bekommen Sie für die angegebene TAC-Klasse die neue und alte Prozess-Anzahl ausgegeben. Die Ausgabe erfolgt am Administrator-Terminal.

```
TACCLASS      TASKS         TASKS-FREE
             NEW    OLD      NEW    OLD
tac-class    number number   number number
```

Erläuterung zur Ausgabe

TACCLASS	Nummer der TAC-Klasse
TASKS	absolute Anzahl der Prozesse, die zur Bearbeitung der TACs dieser TAC-Klasse zur Verfügung stehen. Haben Sie KDCTCL ...TASKSFREE= aufgerufen, dann wird für TASKS folgender Wert ausgegeben: derzeit für die Anwendung eingestellte Prozess-Anzahl - TASKSFREE
TASKS-FREE	Anzahl der Prozesse, die für andere TAC-Klassen freigehalten werden. Wenn Sie KDCTCL ...TASKS= angegeben haben, steht in der Ausgabe unter TASKS-FREE immer 0, um zu zeigen, dass Sie für diese TAC-Klasse eine absolute Angabe gemacht haben.

Beispiel

Die folgende Tabelle zeigt die Auswirkungen verschiedener Änderungen in den Prozess-Anzahlen:

	Dialog-TACs			**Asynchrone TACs**		
Aktion	**CURRENT TASKS**	**TASKS FREE**	**TASKS**	**CURRENT TASKS**	**TASKS FREE**	**TASKS**
Anfangszustand	4	0	3	3	0	3
Änderung TASKS-FREE 0 -->2	4	2	2	3	2	1
Änderung CURRENT TASKS um 2 verringert	2	2	1	1	2	0

CURRENT-TASKS
ist die maximale Anzahl der Prozesse, die derzeit gleichzeitig für die Anwendung eingesetzt werden können (Dialog-TACs) bzw.
die maximale Anzahl der Prozesse, die gleichzeitig Asynchron-Aufträge bearbeiten dürfen (Asynchron-TACs).

TASKS
bezeichnet die jeweils maximale Anzahl der Prozesse für die angegebene TAC-Klasse.

TASKS-FREE
bezeichnet die Anzahl der Prozesse, die für andere TAC-Klassen freigehalten werden.

12.21 KDCUSER - Benutzereigenschaften ändern

Mit KDCUSER können Sie:

- Benutzerkennungen der Anwendung sperren oder wieder zulassen
- Passwörter für Benutzerkennungen definieren, ändern oder löschen.

Wirkung in UTM-Cluster-Anwendungen

KDCUSER wirkt in UTM-Cluster-Anwendungen Cluster-global.

Wirkungsdauer der Änderungen

Änderungen bleiben über das Beenden der Anwendung hinaus gültig.

```
KDCUSER     USER={ username | (username_1,username_2,...,username_10) }
            [ ,PASS=password ]
            [ ,STATUS={ ON| OFF } ]
```

Für die Administration über Message Queuing müssen Sie KDCUSERA angeben.

USER=(user1,user2,...)

Namen der zu administrierenden Benutzerkennungen. Sie können pro Aufruf maximal 10 Namen angeben. Bei nur einem Namen können die Klammern entfallen.

PASS=password

Passwort für die Benutzerkennung neu vergeben, ändern oder löschen.

Das Passwort kann bis zu 16 Zeichen lang sein. Ist das angegebene Passwort kürzer als 16 Zeichen, dann füllt openUTM mit Leerzeichen auf.

Sie können das Passwort als hexadezimale Zeichenfolge (32 Halb-Bytes) in der Form X'.....' oder als Character-String C'....' angeben.

Beispiel:
hexadezimal Zeichenfolge: X'F1F2F3F4F5F6F7F8F9F0'
Character-String: 'ABCDEFGHIJKLMNOP'

Sie löschen ein Passwort durch Angabe von PASS=C' ' (Leerzeichen). Die Eingabe von 16 Zeichen binär Null
(X'00000000000000000000000000000000')
bewirkt keine Änderung des Passworts.

Passwörter löschen können Sie nur, wenn:

- die beim Eintragen der Benutzerkennung festgelegte Minimallänge des Passworts gleich 0 ist
- für die Benutzerkennung keine besondere Komplexitätsstufe definiert ist (NONE).

Ist für eine Benutzerkennung ein Passwort mit begrenzter Gültigkeitsdauer generiert, dann können Sie bei der Passwortänderung als neues Passwort nicht das alte Passwort eingeben.

Ist die Anwendung mit SIGNON GRACE=NO generiert, so ist die generierte Gültigkeitsdauer vom Zeitpunkt der Änderung auch für das neue Passwort wirksam.
Ist die Anwendung mit SIGNON GRACE=YES und das Passwort mit begrenzter Gültigkeitsdauer generiert, so ist das Passwort sofort ungültig.
Wird ein Passwort mit begrenzter Gültigkeitsdauer gelöscht, dann ist keine Gültigkeitsdauer wirksam. Wird danach ein neues Passwort vergeben, ist die Gültigkeitsdauer wieder wirksam.

STATUS=

ON Benutzerkennung wieder zulassen

OFF Benutzerkennung sperren. Die Sperre wirkt beim nächsten Anmeldeversuch dieses Benutzers. Die Funktion wirkt nicht für den Administrator.

Ausgabe von KDCUSER

Am Administrator-Terminal werden der neue und der alte Status der administrierten Benutzerkennungen angezeigt, sowie ggf. der Hinweis, dass das Passwort geändert wurde.

```
USER          STATUS
           NEW      OLD
user1     ON|OFF   ON|OFF         PASSWORD CHANGED
```

13 Message Queues administrieren, Druckausgabe steuern

Um Message Queues zu administrieren und Druckerausgaben zu steuern, haben Sie zwei Möglichkeiten:

1. über die Programmschnittstelle KDCS mit den Funktionen DADM (**D**elayed free message **adm**inistration) und PADM (**P**rinter **adm**inistration) oder
2. über WinAdmin oder WebAdmin, die Ihnen die Funktionalität von DADM und PADM über eine grafische Bedienoberfläche zur Verfügung stellen.

In den folgenden Abschnitten sind die Möglichkeiten beschrieben, wie Sie die Funktionen DADM und PADM nutzen können. Die hier genannten Voraussetzungen und Bedingungen gelten gleichermaßen für die Administration über WinAdmin oder WebAdmin.

Mit DADM können Sie Aufträge und Nachrichten administrieren, die in lokalen Message Queues zwischengespeichert sind und zur Bearbeitung anstehen. Die Message Queues in openUTM sind, mit Ausnahme der Dead Letter Queue, Empfänger-spezifisch, d.h. in einer Queue stehen alle Aufträge, die an dasselbe Ziel gerichtet sind. Empfänger können beispielsweise sein: TACs der eigenen oder einer fernen Anwendung (in diesen Queues stehen Hintergrundaufträge), LTERM-Partner von Terminals, TS-Anwendungen oder Druckern (in diesen Queues stehen Ausgabe-Aufträge), Benutzerkennungen und temporäre Queues. Die Dead Letter Queue ist eine TAC-Queue und enthält nicht ordnungsgemäß verarbeitete Nachrichten unterschiedlicher Empfänger. Weitere Informationen zu Message Queues finden Sie im Abschnitt „Message Queues administrieren (DADM)" und ausführliche Informationen im openUTM-Handbuch „Konzepte und Funktionen“.

Mit PADM können Sie die Ausgabe der Asynchron-Nachrichten an Druckern steuern, d.h. die Druckausgabe beeinflussen, und die Drucker selbst administrieren. Voraussetzung für die Druckeradministration und Drucksteuerung über PADM-Funktionen ist, dass der Drucker einem Druckersteuer-LTERM zugeordnet ist (siehe Abschnitt "Druckersteuer-LTERMs - Administration „lokaler Drucker“" im Abschnitt "Berechtigungskonzept (BS2000-, Unix- und Linux-Systeme)").

Mit DADM können Sie Teilprogramme erstellen, die folgende Funktionen ausführen:

- Informationen über die in einer Message Queue zwischengespeicherten Aufträge und Nachrichten ausgeben.
- Einen Auftrag oder eine Nachricht in der Queue vorziehen, so dass er vor allen anderen Aufträgen der Queue bearbeitet wird.
- Einen Auftrag oder eine Nachricht stornieren, d.h. aus der Queue löschen.
- Nachrichten aus der Dead Letter Queue verschieben, um sie verarbeiten zu können.

Mit PADM kann ein Teilprogramm folgende Funktionen zur Steuerung von Druckausgaben ausführen:

- Einen speziellen Quittungsmodus ein- und ausschalten, bei dem jede Druckausgabe bestätigt werden muss, bevor der nächste Ausgabe-Auftrag bearbeitet wird.
  Diese Aktion wirkt in UTM-Cluster-Anwendungen Cluster-global.
- Druckausgaben wiederholen, z.B. nach einem erfolgreichen Probedruck. Voraussetzung ist, dass der Quittungsmodus eingeschaltet ist.
  Diese Aktion wirkt in UTM-Cluster-Anwendungen Knoten-lokal.
- Eine Liste mit den noch zu quittierenden Druckausgaben ausgeben.
  Diese Aktion wirkt in UTM-Cluster-Anwendungen Knoten-lokal.

Mit PADM kann ein Teilprogramm folgende Funktionen zur Administration von Druckern ausführen:

- Einen Drucker sperren und wieder freigeben. Diese Aktion wirkt in UTM-Cluster-Anwendungen Cluster-global.
- Die Verbindung zu einem Drucker auf- oder abbauen. Diese Aktion wirkt in UTM-Cluster-Anwendungen Knoten-lokal.
- Die Zuordnung Drucker zu LTERM-Partner ändern, z.B. bei Ausfall eines Druckers kann der LTERM-Partner dieses Druckers zusammen mit der zugehörigen Message Queue einem anderen Drucker zugeordnet werden, der dann die in der Queue stehenden Druckaufträge bearbeitet.
  Diese Funktion ist nur in stand-alone UTM-Anwendungen erlaubt.
- Drucker zu Druckerbündeln zusammenfassen. Dabei ordnen Sie einem LTERM-Partner mehrere Drucker zu. Die Message Queue des LTERM-Partners wird dann von allen Druckern des Druckerbündels gemeinsam abgearbeitet. Zu Druckerbündeln siehe auch openUTM-Handbuch „Anwendungen generieren“.
  Diese Funktion ist nur in stand-alone UTM-Anwendungen erlaubt, in UTM-Cluster-Anwendungen können Druckerbündel nur statisch generiert werden.
- Informationen über Drucker ausgeben.

Für die Administration von Druckern mit PADM-Aufrufen ist die UTM-Administrationsberechtigung nicht unbedingt erforderlich. Welche Berechtigung Sie benötigen, um Teilprogramme mit DADM- und PADM-Aufrufen starten zu können, ist im Abschnitt „Berechtigungskonzept (BS2000-, Unix- und Linux-Systeme)" beschrieben.

Mit openUTM werden die Beispiel-Teilprogramme KDCDADM und KDCPADM ausgeliefert, die die Funktionen von DADM und PADM nutzen. Mit diesen Teilprogrammen können Sie Asynchron-Aufträge administrieren und Druckausgaben sowie Drucker steuern, ohne selbst Teilprogramme schreiben zu müssen. In der folgenden Beschreibung wird mit folgendem Symbol auf die entsprechende Funktion von KDCDADM bzw. KDCPADM verwiesen:

Wenn Sie mit PADM und DADM eigene Teilprogramme erstellen, haben Sie jedoch die Möglichkeit, die Benutzeroberfläche in dem Teilprogramm selbst zu gestalten, z.B. auf BS2000-Systemen durch Eingabe über Formate.

Die Beschreibung der Aufrufe DADM und PADM finden Sie im openUTM-Handbuch „Anwendungen programmieren mit KDCS“. Die Teilprogramme KDCDADM und KDCPADM sind im Abschnitt „UTM-Teilprogramme für DADM- und PADM-Funktionen" beschrieben.

Damit Sie die Beispielprogramme KDCDADM und KDCPADM oder eigene Teilprogramme mit PADM- oder DADM-Aufrufen nutzen können, müssen Sie die Teilprogramme statisch oder dynamisch in die Konfiguration der Anwendung aufnehmen und ihnen Transaktionscodes zuordnen.

i openUTM auf Windows-Systemen unterstützt keine Drucker. Der KDCS-Aufruf PADM und das Teilprogramm KDCPADM stehen zwar zur Verfügung, sind aber für UTM-Anwendungen auf Windows-Systemen nicht relevant. Für alle Aktionen, die Sie mit DADM bzw. KDCDADM durchführen, ist Administrationsberechtigung erforderlich.

13.1 Berechtigungskonzept (BS2000-, Unix- und Linux-Systeme)

PADM und DADM sind keine Funktionen der Programmschnittstelle zur Administration. Deshalb gilt für die Vorgänge, die PADM und DADM nutzen, ein anderes Berechtigungskonzept. Dieses Berechtigungskonzept ermöglicht es, dass auch Benutzer ohne Administrationsberechtigung ihre Ausgabe-Aufträge an den „lokalen“ Drucker selbst administrieren können. Auch die Administration des „lokalen“ Druckers kann der Benutzer vornehmen, ohne eine besondere Berechtigung zu haben.

Dazu müssen Sie für die Drucker Druckersteuer-LTERMs erzeugen und den Druckern zuordnen, die „lokal“ administriert werden sollen, d.h. von einem Benutzer/Client ohne Administrationsberechtigung. Über das Druckersteuer-LTERM können dann die zugehörigen Drucker und deren Queues von jedem Benutzer oder Client, der sich über das Druckersteuer-LTERM anschließt, administriert werden.

Für folgende Administrationsaufgaben ist jedoch die Administrationsberechtigung erforderlich:

- Administration von Hintergrund-Aufträgen und Ausgabe-Aufträgen für Terminals oder ferne TS-Anwendungen.
- Administration von Ausgabe-Aufträgen und Druckern über einen beliebigen LTERM-Partner.

  Ein Benutzer, der die UTM-Administrationsberechtigung hat, kann alle Drucker an allen Druckersteuer-LTERMs und alle Asynchron-Aufträge administrieren, unabhängig davon, über welchen LTERM-Partner die Vorgänge gestartet werden.
- Administration von Service-gesteuerten Queues (USER-, TAC- und temporäre Queues).

Druckersteuer-LTERMs - Administration „lokaler Drucker“

Ein Druckersteuer-LTERM ist ein LTERM-Partner, der als Dialog-Partner eingetragen ist (*usage_type*=D). Über diesen LTERM-Partner kann sich ein Client oder ein Terminal-Benutzer an die Anwendung anschließen. Von dem Terminal bzw. Client aus kann dann die Administration der Drucker und der zugehörigen Queues erfolgen, die dem Druckersteuer-LTERM zugeordnet sind.

Drucker werden dem Druckersteuer-LTERM wie folgt zugeordnet:
Jedem Drucker wird ein LTERM-Partner, der als Ausgabemedium konfiguriert ist (*usage_type*='O'), zugeordnet. Alle Ausgabe-Aufträge für diesen Drucker „schickt“ openUTM an den LTERM-Partner des Druckers, d.h. openUTM schreibt die Ausgabe-Aufträge in die Message Queue des LTERM-Partners, das ist die Queue des zugehörigen Druckers. Einem LTERM-Partner können Sie auch mehrere Drucker zuordnen (Druckerbündel). Dann arbeiten alle diese Drucker mit einer Queue.

Die LTERM-Partner der Drucker ordnen Sie den Druckersteuer-LTERMs zu.

Dazu müssen Sie beim Erzeugen des LTERM-Partners in CTERM/*kc_lterm_str.cterm* (CTERM=**C**ontrol **TERM**inal) angeben, zu welchem Druckersteuer-LTERM der zugehörige Drucker gehören soll. In CTERM/*kc_lterm_str*.*cterm* geben Sie den Namen des Druckersteuer-LTERMs an (Name des LTERM-Partners).

Einem Druckersteuer-LTERM können Sie Einzeldrucker und auch Druckerbündel zuordnen. Für jeden Drucker, der einem Druckersteuer-LTERM zugeordnet wird, müssen Sie eine Drucker-Id definieren. Diese Drucker-Id muss im Bereich des Druckersteuer-LTERMs eindeutig sein, da das Druckersteuer-LTERM die Drucker über die Drucker-Id anspricht. Auf die Eindeutigkeit der Drucker-Id müssen Sie insbesondere bei Druckerbündeln achten. Sie müssen für jeden Drucker des Bündels eine eigene Drucker-Id definieren. Die Drucker-Id müssen Sie dem Drucker beim Eintragen in die Konfiguration zuordnen.

Ein Beispiel für die Konfiguration mit KDCDEF ist im Bild dargestellt.

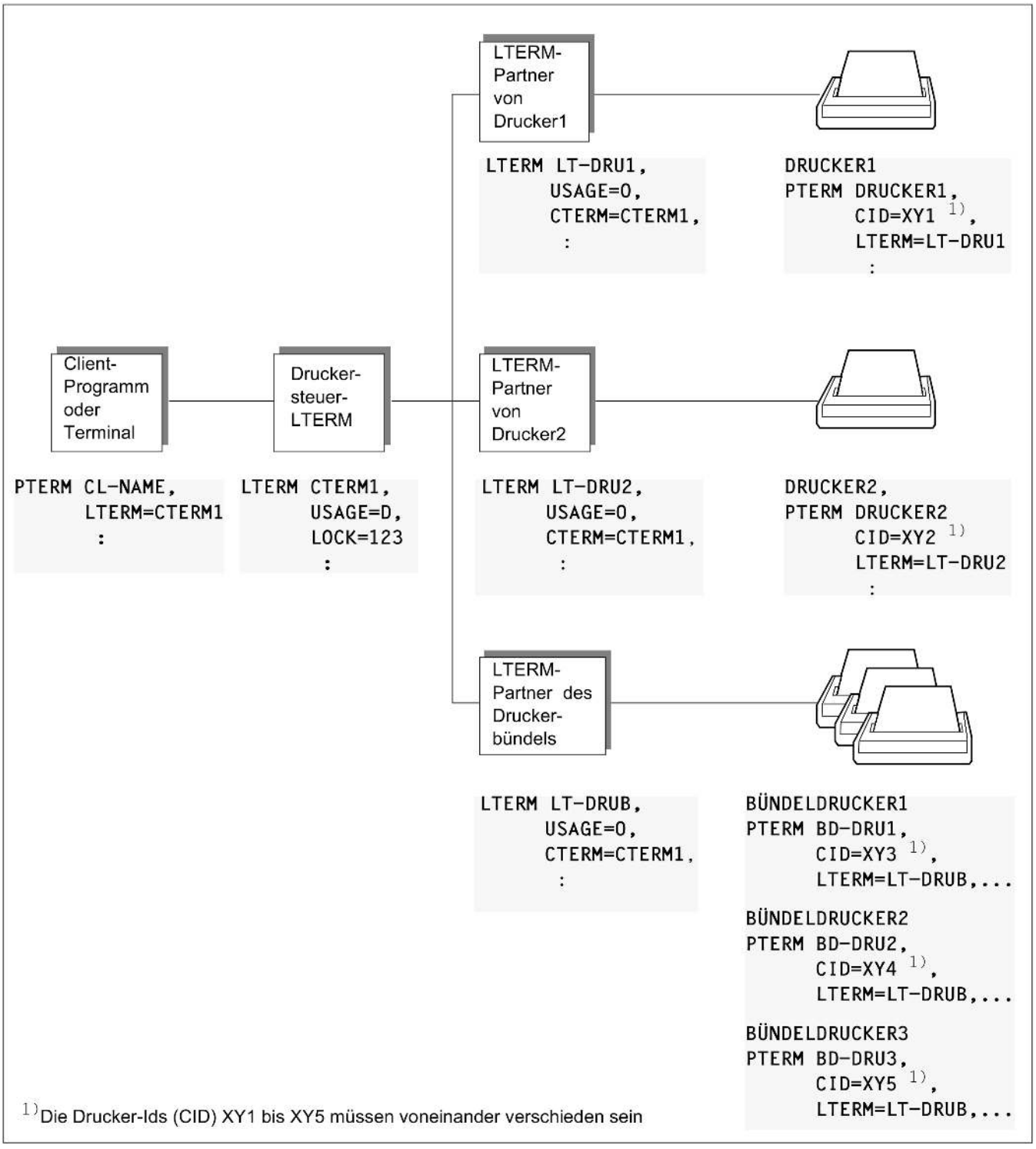
Um den Zugang zum Druckersteuer-LTERM auf einen bestimmten Personenkreis zu beschränken, können Sie dem Druckersteuer-LTERM einen Lockcode zuordnen. Entsprechend können Sie auch die Transaktionscodes der

PADM- und DADM-Teilprogramme über Lockcodes oder Access Listen schützen. Damit können Sie festlegen, welcher Benutzer/Client welche Administrationsfunktionen ausführen darf. Dem Druckersteuer-LTERM sollten Sie aber auf jeden Fall alle Keycodes für die Teilprogramme zur Druckeradministration und Drucksteuerung zuordnen (zum Lock-/Keycode-Konzept siehe openUTM-Handbuch „Konzepte und Funktionen“).

Über ein Druckersteuer-LTERM kann ein Benutzer/Client Vorgänge starten, die:

- die zugehörigen Drucker über PADM-Aufrufe administrieren.
- die an den Drucker gerichteten Ausgabe-Aufträge administrieren (DADM-Aufrufe).
- die Druckausgabe an diesen Druckern steuern.

Teilprogramme, die DADM- und PADM-Funktionen nutzen und über ein Druckersteuer-LTERM gestartet werden sollen, müssen Sie als Dialog-Programme schreiben und ihnen Dialog-TACs zuordnen.



Erzeugen eines Druckersteuer-LTERMs und der zugehörigen Drucker

13.2 Message Queues administrieren (DADM)

Mit DADM können Sie zwei unterschiedliche Arten von Message Queues administrieren. Das sind:

- UTM-gesteuerte Queues

  Die von einem Teilprogramm erzeugten Asynchron-Aufträge werden dem Empfänger zum vorgegebenen Zeitpunkt zugestellt. Für Nachrichten an TACs wird dazu das zugehörige Teilprogramm durch openUTM gestartet.

- Service-gesteuerte Queues

  Bei diesen Queues wird die Verarbeitung nicht durch UTM, sondern durch das Teilprogramm selbst gesteuert.

  Für Service-gesteuerte Queues stehen drei Typen zur Verfügung:

  1. USER-Queues

     Jedem Benutzer einer UTM-Anwendung steht automatisch unter seiner Benutzerkennung eine permanente Queue zur Verfügung. Eine solche Queue ist über die Benutzerkennung erreichbar. USER-Queues bieten die Möglichkeit, beispielsweise asynchrone Nachrichten an einen UPIC-Benutzer zu senden.

  2. TAC-Queues

     Durch die Generierung von TACs vom Typ 'Q' werden permanente Queues mit festem Namen erzeugt. Über eine solche Queue lassen sich beispielsweise Queues in fernen UTM-Anwendungen realisieren, die von der lokalen UTM-Anwendung über einen LTAC-Namen angesprochen werden.

     Die Dead Letter Queue KDCDLETQ ist eine TAC-Queue, die immer zur Verfügung steht, um Nachrichten zu sichern, die nicht verarbeitet werden konnten.

  3. Temporäre Queues

     Temporäre Queues können dynamisch erzeugt und gelöscht werden. Der Name einer solchen Queue kann von dem Teilprogramm oder implizit durch openUTM erzeugt werden. Mit Hilfe temporärer Queues ist z.B. die Kommunikation zwischen zwei Vorgängen möglich: Ein Vorgang richtet die Queue ein und sendet eine Nachricht an die Queue; ein anderer Vorgang liest die Nachricht und löscht anschließend die Queue.

     Die maximal möglich Anzahl temporärer Queues wird mit der Generierungsanweisung QUEUE festgelegt.

     Zum Erzeugen und Löschen von temporären Queues stehen Ihnen die KDCS-Aufrufe QCRE und QREL zur Verfügung. Diese Aufrufe sind im openUTM-Handbuch „Anwendungen programmieren mit KDCS“ beschrieben.

Die Administration von Nachrichten in einer Queue erfolgt über die Programmschnittstelle KDCS mit DADM. Mit FPUT und DPUT können Sie Hintergrund-Aufträge, Ausgabe-Aufträge und Nachrichten für Service-gesteuerte Queues erzeugen. Welche Funktion DADM im einzelnen durchführt, ist abhängig von der Operationsmodifikation, die Sie im Feld *kcom* des Parameterbereichs an openUTM übergeben. Folgende Operationsmodifikationen stehen zur Verfügung:

- DADM RQ (**R**ead **Q**ueue) zum Lesen von Informationen über die Nachrichten in einer Message Queue.
- DADM UI (**U**ser **I**nformation) zum Lesen von Benutzerinformationen zu einer Nachricht. Benutzerinformationen werden beim Erzeugen einer Nachricht vom Auftraggeber geschrieben und im angegebenen Empfangsbereich übergeben.
- DADM CS (**C**hange **S**equence) ändert die Reihenfolge der Nachrichten in einer Queue. Sie können damit eine Nachricht von einer beliebigen Stelle in der Queue an die erste Stelle vorziehen. Diese Nachricht wird dann vor allen anderen Nachrichten der Queue bearbeitet.
- DADM DL (**D**e**l**ete) und DADM DA (**D**elete **A**ll) zum Löschen einer einzelnen Nachricht oder aller Nachrichten einer Queue.

Beim Löschen von Auftrags-Komplexen mit DADM DL können Sie negative Quittungsaufträge aktivieren. Ein Auftrags-Komplex ist ein Asynchron-Auftrag mit positivem und/oder negativem Quittungs-Auftrag (siehe openUTM-Handbuch „Anwendungen programmieren mit KDCS“, MCOM-Aufruf).

Beim Löschen von Nachrichten mit DADM DA werden die Nachrichten einschließlich der Folgenachrichten gelöscht. Ein solcher Löschaufruf wird nur ausgeführt

- bei UTM-gesteuerten Queues, wenn für das angegebene Ziel kein Auftrag in Bearbeitung ist,
- bei Service-gesteuerten Queues, wenn gerade keine Nachricht gelesen wird.

- DADM MV (**M**o**v**e) und DADM MA (**M**ove **a**ll) zum Verschieben einer oder aller Nachrichten, die in der Dead Letter Queue gespeichert wurden. Die Nachrichten können der jeweiligen ursprünglichen Message Queue oder einem beliebigen Ziel vom gleichen Typ (Asynchron-TAC/TAC-Queue, LPAP-Partner, OSI-LPAP-Partner) zugeordnet werden.

Damit openUTM DADM-Nachrichten bearbeiten kann, müssen Sie die Message Queue und die Nachricht in der Queue eindeutig identifizieren.

Identifikation der Message Queue

Die Message Queues in openUTM sind Empfänger-spezifisch, d.h. für jeden Empfänger von Aufträgen bzw. Nachrichten verwaltet entweder openUTM oder das Teilprogramm selbst eine eigene Message Queue. Eine zu administrierende UTM-gesteuerte Message Queue ist eindeutig identifiziert, wenn Sie beim Aufruf von DADM den Namen des Empfängers angeben. Bei UTM-gesteuerten Queues geben Sie beispielsweise an:

- bei Ausgabe-Aufträgen den Namen des LTERM-Partners, dem das Terminal, der Drucker oder die TS-Anwendung zugeordnet ist,
- bei Hintergrund-Aufträgen den Namen des Asynchron-TACs, an den der Auftrag gerichtet ist.

Bei Service-gesteuerten Queues ist zur Identifikation der Name und der Typ der Queue erforderlich.

Den Namen des Empfängers übergeben Sie bei DADM RQ/DL/DA im Feld *kclt*, den Typ im Feld *kcqtyp* des KB-Parameterbereichs.

Identifikation von Nachrichten in einer Message Queue

openUTM baut für jede Nachricht intern eine eigene Identifikation auf, auch Auftrags-Id oder DPUTID genannt. Dadurch können Sie jede einzelne Nachricht gezielt administrieren.

Nach der Bearbeitung einer Nachricht durch den Empfänger oder Löschen der Nachricht durch die Administration wird die Auftrags-Id freigegeben und kann von openUTM direkt wieder für eine andere Nachricht vergeben werden. Deshalb müssen Sie bei DADM UI/CS/DL-Aufrufen zur eindeutigen Identifikation der zu administrierenden Nachricht zusätzlich den Zeitpunkt der Nachrichtenerzeugung angeben. Nur so kann z.B. verhindert werden, dass eine falsche Nachricht mit DADM DL storniert wird.

Auftrags-Id und Zeitpunkt der Nachrichtenerzeugung müssen Sie bei den DADM-Aufrufen im KB-Parameterbereich übergeben. Beides können Sie mit DADM RQ ermitteln und in folgenden DADM-Aufrufen verwenden.

> **i** Werden die in der KDCFILE zwischengespeicherten Nachrichten (FPUT- und DPUT-Nachrichten) mit dem UTM-Tool KDCUPD in eine neue KDCFILE übertragen, dann erhalten sie **neue** Auftrags-Ids.

13.2.1 Informieren über Nachrichten in einer Queue - DADM RQ

Mit dem Aufruf DADM RQ liefert openUTM Informationen über die Nachrichten in einer Queue. openUTM liefert zu jeder Nachricht die Auftrags-Id, die Benutzerkennung des Auftraggebers, Entstehungszeit der Nachricht und bei zeitgesteuerten Nachrichten (DPUT-Nachrichten) den frühesten Ausführungszeitpunkt sowie die Information, ob ein positiver oder ein negativer Quittungs-Auftrag vorhanden ist.

Beim Aufruf von DADM RQ geben Sie im Feld *kclt* des KB-Parameterbereichs den Namen des Empfängers an, dessen Message Queue gelesen werden soll. Bei Service-gesteuerten Queues ist auch der Typ im Feld *kcqtyp* erforderlich.

Sie können sich Informationen über alle Nachrichten in einer Message Queue ausgeben lassen oder die Informationen auf eine Nachricht in der Queue beschränken.

Im Feld *kcrn* des Parameterbereichs geben Sie dann die Auftrags-Id der Nachricht an, über die openUTM informieren soll. Falls Sie Leerzeichen in *kcrn* schreiben, dann informiert openUTM über die erste Nachricht in der Message Queue des Empfängers im Feld *kclt*.

Das Lesen der Informationen über alle Nachrichten einer Message Queue läuft wie folgt ab:

- Beim ersten DADM RQ für einen Empfänger geben Sie statt einer Auftrags-Id Leerzeichen im Feld *kcrn* des Parameterbereichs an.
- openUTM liefert Informationen über die erste Nachricht in der Message Queue des Empfängers zurück. Existiert mindestens eine weitere Nachricht an denselben Empfänger, dann schreibt openUTM die Auftrags-Id der nächsten Nachricht der Queue in das Feld *kcrmf* des KB-Rückgabebereichs.
- Sie rufen DADM RQ erneut auf und schreiben die Auftrags-Id, die openUTM im Feld *kcrmf* zurückgeliefert hat, in das Feld *kcrn* des KB-Parameterbereichs.
- openUTM liefert Informationen zu der zweiten Nachricht und die Auftrags-Id der in der Queue folgenden Nachricht zurück, sofern eine weitere Nachricht existiert.

So kann die Message Queue der Reihe nach abgearbeitet werden. Beim Lesen der Informationen zur letzten Nachricht der Queue liefert openUTM im Feld *kcrmf* Leerzeichen zurück.

Für die bei DADM RQ zurückgegebenen Informationen gibt es eine Datenstruktur, die Sie über den Nachrichtenbereich legen können. Die C-Datenstruktur heißt *kc_dadm* und gehört zum Include *kcdad.h*, die COBOL-Datenstruktur heißt KCDADC.

KDCDADM INFORM im Abschnitt "INFORM - Über Message Queues und Nachrichten informieren"

13.2.2 Benutzerinformation zu einer Nachricht lesen - DADM UI

Oft reichen die Informationen, die openUTM über Nachrichten zur Verfügung stellt (siehe Abschnitt "Informieren über Nachrichten in einer Queue - DADM RQ"), nicht aus, damit der Administrator eine Nachricht eindeutig identifizieren kann. Deshalb kann der Auftraggeber beim Erstellen einer Nachricht mit dem DPUT-Aufruf zusätzliche Informationen - *Benutzerinformationen* genannt - hinterlegen. Benutzerinformationen schreibt man mit dem Aufruf DPUT NI bzw. bei Quittungsaufträgen in Auftrags-Komplexen mit DPUT +I oder DPUT -I (siehe openUTM-Handbuch „Anwendungen programmieren mit KDCS“).
Die Benutzerinformationen werden nicht zum Empfänger der Nachricht übermittelt. Sie sind aber über die Auftrags-Id an die Nachricht gekoppelt und können nur mit DADM UI gelesen werden.

Beim Aufruf von DADM UI müssen Sie die Auftrags-Id und den Zeitpunkt der Nachrichtenerzeugung im KB-Parameterbereich übergeben. Beides kann zuvor mit DADM RQ ermittelt werden.

Die Benutzerinformation zu Quittungsaufträgen in Auftrags-Komplexen können Sie erst lesen, wenn der Quittungs-Auftrag aktiviert wurde.

KDCDADM INFORM, LIST=LONG im Abschnitt "INFORM - Über Message Queues und Nachrichten informieren"

13.2.3 Nachrichten in der Queue vorziehen - DADM CS

Der Aufruf DADM CS ist dann sinnvoll, wenn zu einem Zeitpunkt mehrere Nachrichten für denselben Empfänger zur Bearbeitung anstehen. Durch DADM CS wird die angegebene Nachricht, identifiziert durch ihre Auftrags-Id und den Zeitpunkt der Erzeugung der Nachricht, an die erste Stelle der Message Queue vorgezogen. Auftrags-Id und den Zeitpunkt der Erzeugung der Nachricht können Sie mit DADM RQ ermitteln.

Beachten Sie, dass zeitgesteuerte Nachrichten nur vorgezogen werden können, wenn der bei der Nachrichtenerzeugung mit DPUT angegebene „früheste Ausführungszeitpunkt“ schon erreicht ist. Andernfalls lehnt openUTM den DADM CS-Aufruf ab (Returncode 40Z).

KDCDADM NEXT im Abschnitt "NEXT - Nachrichten in der Message Queue vorziehen"

13.2.4 Nachrichten aus einer Queue löschen - DADM DA/DL

Mit DADM DA löschen Sie alle noch nicht bearbeiteten Nachrichten an einen bestimmten Empfänger. Bei Service-gesteuerten Queues können Nachrichten, die gerade gelesen werden, nicht gelöscht werden. Wird eine Service-gesteuerte Queue dynamisch gelöscht (KC_DELETE_OBJECT bzw. QREL RL), dann gehen auch die Nachrichten in dieser Queue verloren. Nachrichten, die zum Zeitpunkt des DADM DA-Aufrufs bereits vom Empfänger bearbeitet werden, werden nicht gelöscht. Den Empfänger müssen Sie beim DADM DA-Aufruf im Feld *kclt* des KB-Parameterbereichs angeben.

Mit DADM DL löschen Sie eine bestimmte Nachricht. Zur Identifikation der Nachricht müssen Sie deren Auftrags-Id und den Zeitpunkt angeben, an dem die Nachricht erzeugt wurde. Beides kann mit DADM RQ ermittelt werden.

Wird die angegebene Nachricht bereits vom Empfänger bearbeitet, dann wird der DADM DL-Aufruf von openUTM abgelehnt (Returncode 40Z).

Insbesondere können Sie mit DADM DA/DL keine bereits gestartete Druckausgabe löschen. Dazu müssen Sie wie folgt vorgehen:

1. Die Verbindung zum Drucker, an dem der Auftrag bearbeitet wird, abbauen (PADM CS). openUTM baut die Verbindung zum Drucker auch ab, wenn Sie den Drucker mit PADM CS sperren.
2. Den Druck-Auftrag löschen (DADM DL).
3. Die Verbindung zum Drucker wieder aufbauen (PADM CS; siehe Abschnitt "Druckerstatus ändern - PADM CS").

Sind der zu löschenden Nachricht Quittungsaufträge zugeordnet (DPUT-Aufträge in Auftrags-Komplexen), dann kann man bei DADM DL angeben, ob beim Löschen der negative Quittungs-Auftrag aktiviert werden soll oder die Quittungsaufträge zusammen mit dem Hauptauftrag gelöscht werden sollen (Feld *kcmod* des KB-Parameterbereichs).

Zu Auftrags-Komplexen und Quittungsaufträgen siehe openUTM-Handbuch „Anwendungen programmieren mit KDCS“.

KDCDADM DELETE im Abschnitt "DELETE - Nachricht aus Message Queue löschen"

13.2.5 Nachrichten der Dead Letter Queue verschieben - DADM MA/MV

Die Dead Letter Queue besteht aus Nachrichten, die nicht verarbeitet werden konnten und für die keine Redelivery erfolgt ist. Um diese Nachrichten evtl. nach einer Fehlerbehebung noch verarbeiten zu können, müssen sie entweder ihrem ursprünglichen Ziel oder einem neuen Ziel zugeordnet werden.

Mit DADM MA verschieben Sie mehrere Nachrichten, die in der Dead Letter Queue gespeichert wurden. Die Nachrichten können der jeweiligen ursprünglichen Message Queue oder einem beliebigen neuen Ziel vom gleichen Typ (Asynchron-TAC/TAC-Queue, LPAP-Partner, OSI-LPAP-Partner) zugeordnet werden. Geben Sie ein neues Ziel an, so werden nur die Nachrichten mit passendem ursprünglichem Ziel (d.h. gleicher Typ) verschoben.

Mit DADM MV verschieben Sie eine einzelne Nachricht der Dead Letter Queue. Zur Identifikation der Nachricht müssen Sie die Auftrags-Id und den Zeitpunkt der Erzeugung der Nachricht angeben.

Zur Identifikation des Ziels geben Sie an:

- den TAC, wenn die Nachrichten mit ursprünglichem Ziel TAC oder TAC-Queue an ein Asynchron-Programm gerichtet werden sollen,
- den Namen einer TAC-Queue, wenn die Nachrichten mit ursprünglichem Ziel TAC oder TAC-Queue an eine Service-gesteuerte Queue gerichtet werden sollen,
- den Namen eines LPAP-Partners (aber kein Master-LU61-LPAP), wenn die Nachrichten mit ursprünglichem Ziel LPAP an einen LPAP-Partner gerichtet werden sollen,
- den Namen eines OSI-LPAP-Partners (aber kein Master-OSI-LPAP), wenn die Nachrichten mit ursprünglichem Ziel OSI-LPAP an einen OSI-LPAP-Partner gerichtet werden sollen,
- Leerzeichen, wenn die Nachrichten wieder ihrem jeweiligen ursprünglichen Ziel zugeordnet werden sollen.

Bei DADM MA mit KCLT= Leerzeichen verbleiben Nachrichten, deren ursprüngliches Ziel nicht mehr existiert, in der Dead Letter Queue. Diesen Nachrichten können Sie Asynchron-Transaktionscodes oder TAC-Queues als neue Ziele zuordnen.

Beim Verschieben von Nachrichten aus der Dead Letter Queue wird ein evtl. definiertes QLEV und der STATUS der Empfänger-Queue ignoriert. Beim Verschieben kann also der Queue-Level überschritten und es können Nachrichten an gesperrte TACs gesendet werden.

> Das ursprüngliche Ziel einer Nachricht der Dead Letter Queue kann der Rückgabeinformation des DADM RQ Aufrufs entnommen werden.

KDCDADM MOVE im Abschnitt "MOVE - Nachrichten aus der Dead Letter Queue verschieben"

13.3 Drucker administrieren und Druckausgaben steuern (PADM)

Mit dem KDCS-Aufruf PADM können Sie Teilprogramme erstellen, die die Ausgabe von Asynchron-Nachrichten am Drucker steuern und Drucker administrieren. Mit PADM-Funktionen können nur Drucker administriert werden, die einem Druckersteuer-LTERM zugeordnet sind.

Identifikation von Druckern bei der Administration über PADM-Aufrufe

Teilprogramme, die die Druckausgabe steuern und Drucker administrieren sollen, müssen die Drucker eindeutig identifizieren können. Um hierbei vom Druckernamen unabhängig zu sein, müssen Sie für jeden Drucker, der einem Druckersteuer-LTERM zugeordnet wird, eine Drucker-Id definieren. Die Drucker-Id wird beim Eintragen des Druckers in die Konfiguration festgelegt. Die Drucker-Id muss im Bereich des Druckersteuer-LTERMs eindeutig sein.

Ein Drucker ist dann Anwendungs-weit eindeutig identifizierbar über den Namen des Druckersteuer-LTERMs, zu dem er gehört, und seine Drucker-Id. Bei der Administration vom Druckersteuer-LTERM aus genügt die Drucker-Id zur Identifikation des Druckers, z.B. beim Bestätigen einer Druckausgabe.

Wenn Sie die Druckausgabe auf einem Drucker steuern wollen, müssen Sie die Drucker-Id des Druckers nicht an das Teilprogramm übergeben. Sie kann innerhalb des Teilproramms mit Hilfe von PADM AI/PI-Aufrufen ermittelt werden.

13.3.1 Drucker administrieren mit PADM

Funktionen zur Druckeradministration bietet openUTM mit dem Aufruf PADM. Welche Funktion PADM im einzelnen durchführt, ist abhängig von der Operationsmodifikation, die Sie im Feld *kcom* des Parameterbereichs an openUTM übergeben. Folgende Operationsmodifikationen stehen zur Verfügung:

- PADM PI (**P**rinter **I**nformation) um Informationen über die Drucker, die einem Druckersteuer-LTERM zugeordnet sind, zu lesen.
- PADM CA (**C**hange **A**ddress) um einen Drucker einem anderen LTERM-Partner zuzuordnen.
- PADM CS (**C**hange **S**tate) zum Ändern des Druckerstatus, d.h. Drucker sperren bzw. wieder zulassen, Verbindung zum Drucker auf- oder abbauen.

13.3.1.1 Informationen über Drucker abfragen - PADM PI

Der Aufruf PADM PI liefert zu jedem Drucker eines Druckersteuer-LTERMs u.a. folgende Informationen zurück:

- Drucker-Id des Druckers
- Name des zugehörigen LTERM-Partners
- Status des Druckers, d.h. openUTM informiert darüber, ob der Drucker z.Zt. gesperrt ist oder ob er mit der Anwendung verbunden ist
- Anzahl der Druckaufträge in der Queue des Druckers
- Anzahl der zeitgesteuerten Druckaufträge in der Queue des Druckers und deren frühester Ausgabezeitpunkt

Diese Informationen können Sie sich z.B. am Druckersteuer-LTERM ausgeben lassen.

Sie können sich Informationen über einen bestimmten Drucker ausgeben lassen. Dazu müssen Sie im Feld *kcrn* des Parameterbereichs dessen Drucker-Id angeben. Übergeben Sie Leerzeichen in *kcrn*, dann informiert openUTM über den ersten Drucker.

Sie können sich aber auch Informationen zu allen Druckern ausgeben lassen, die zu einem Druckersteuer-LTERM gehören. Dabei gehen Sie wie folgt vor:

- Beim ersten PADM PI geben Sie im Feld *kcrn* des Parameterbereichs Leerzeichen an, um die Informationen über den ersten Drucker zu lesen.
- openUTM liefert u.a. die Drucker-Id des ersten Druckers zurück. Existiert mindestens ein weiterer Drucker an diesem Druckersteuer-LTERM, dann schreibt openUTM die Drucker-Id des nächsten Druckers in das Feld *kcrmf* des KB-Rückgabebereichs.
- Sie rufen PADM PI erneut auf und schreiben die Drucker-Id, die openUTM zuvor in *kcrmf* zurückgeliefert hat, in das Feld *kcrn* des KB-Parameterbereichs.
- openUTM liefert Informationen zu dem zweiten Drucker und die Drucker-Id des nächsten Druckers, sofern ein weiterer Drucker existiert, zurück usw.

Beim Lesen der Informationen zum letzten Drucker liefert openUTM im Feld *kcrmf* Leerzeichen zurück.

Für die bei PADM PI zurückgelieferten Informationen gibt es eine Datenstruktur, die Sie über den Nachrichtenbereich legen können. Die C-Datenstruktur heißt *kc_padm* und gehört zum Include *kcpad.h*, die COBOL-Datenstruktur heißt KCPADC.

KDCPADM INFORM, LIST=PRINTERS im Abschnitt "INFORM - Informieren über Drucker eines Druckersteuer-LTERMs"

13.3.1.2 Druckerstatus ändern - PADM CS

Durch den Aufruf PADM CS können Sie die im Folgenden aufgelisteten Aktionen durchführen. Welche Aktion ausgeführt werden soll, legen Sie im Feld *kcact* des Parameterbereichs fest.

- Einen Drucker sperren (*kcact*=OFF) oder einen zuvor gesperrten Drucker wieder freigeben (*kcact*=ON).

  Beide Aktionen wirken in UTM-Cluster-Anwendungen Cluster-gobal.
- Die Verbindung zu einem Drucker aufbauen (*kcact*=CON) oder abbauen (*kcact*=DIS).

Ausgabe-Aufträge an Drucker werden immer in die Message Queue des zugehörigen LTERM-Partners geschrieben. Ist der Drucker gesperrt oder nicht konnektiert, werden die Aufträge solange zwischengespeichert, bis die Sperre wieder aufgehoben bzw. die Verbindung aufgebaut wird, oder Sie dem LTERM-Partner einen anderen nicht gesperrten Drucker zuordnen und diesen konnektieren.

Beim Sperren eines Druckers wird die Verbindung zu ihm automatisch abgebaut und muss nach dem Freigeben explizit wieder aufgebaut werden.

Eine Verbindung zu einem gesperrten Drucker können Sie nicht aufbauen. Soll ein gesperrter Drucker wieder konnektiert werden, müssen Sie wie folgt vorgehen:

1. Sie geben den Drucker wieder frei, dazu rufen Sie PADM CS mit *kcact*=ON auf.
2. Sie informieren sich mit PADM PI darüber, ob openUTM die Sperre aufgehoben hat.
3. Sie rufen PADM CS mit *kcact*=CON auf, um die Verbindung aufzubauen.

Der erste PADM-Aufruf darf nicht in der gleichen Transaktion durchgeführt werden wie die beiden anderen.

KDCPADM STATE im Abschnitt "STATE - Status eines Druckers ändern"

13.3.1.3 Drucker einem anderen LTERM-Partner zuordnen - PADM CA

Mit PADM CA können Sie die Zuordnung Drucker zu LTERM-Partner ändern. Den Namen des neuen LTERM-Partners geben Sie im Feld *kcadrlt* des Parameterbereichs an. Der neue LTERM-Partner, der dem Drucker zugeordnet werden soll, muss bereits in der Konfiguration der Anwendung enthalten und für den Anschluss von Druckern definiert sein (*usage*=O). Dem LTERM-Partner kann bereits ein Drucker zugeordnet sein. Diese alte Zuordnung wird nicht aufgelöst.

Diese Funktion ist nur in stand-alone UTM-Anwendungen erlaubt.

Mit dieser Funktion können Sie deshalb während des Anwendungslaufs Druckerbündel erzeugen, indem Sie mehrere Drucker einem LTERM-Partner zuordnen. Die Queue des LTERM-Partners wird dann von allen Druckern des Druckerbündels abgearbeitet.

Sie können aber auch beim Ausfall eines Druckers den LTERM-Partner des Druckers zusammen mit der Message Queue einem anderen Drucker zuordnen; dieser bearbeitet dann die Ausgabe-Aufträge.

Wird ein Vorgang, der die Zuordnung ändert, von einem nicht-administrationsberechtigten Benutzer oder Client an einem Druckersteuer-LTERM gestartet, dann muss sowohl der LTERM-Partner als auch der Drucker im Zuständigkeitsbereich des Druckersteuer-LTERMs liegen. D.h. dem LTERM-Partner muss dieses Druckersteuer-LTERM zugeordnet sein und der Drucker muss zuvor einen LTERM-Partner gehabt haben, der diesem Druckersteuer-LTERM zugeordnet ist.

PADM CA ist nur erlaubt, wenn der Drucker nicht mit der Anwendung verbunden ist. Dies können Sie vorher mit dem PADM PI-Aufruf überprüfen. Da PADM CA der Transaktionssicherung unterliegt, also erst bei Transaktionsende durchgeführt wird, kann in der Zwischenzeit die Verbindung zum Drucker von einem anderen Vorgang aufgebaut worden sein. Daher sollte man in einer Folge-Transaktion mit PADM PI überprüfen, ob die Aktion tatsächlich durchgeführt wurde.

KDCPADM SWITCH im Abschnitt "SWITCH - Zuordnung Drucker zu LTERM-Partner ändern"

13.3.2 Drucksteuerung mit PADM

Standardmäßig erfolgt die Druckausgabe „ohne“ Drucksteuerung, d.h. openUTM steuert die Ausgabe von Nachrichten an Druckern. Die Druckausgabe erfolgt dann im Automatikmodus. Der Automatikmodus ist nach dem ersten Start der Anwendung eingestellt.

Druckausgabe „mit“ Drucksteuerung bedeutet, dass der Benutzer die Ausgabe von Nachrichten steuern muss. Die Drucksteuerung kann realisiert werden durch:

- Vorgänge mit PADM-Aufrufen, die von einem Druckersteuer-LTERM aus gestartet werden
- Vorgänge mit PADM-Aufrufen, die unter UTM-Administrationsberechtigung ablaufen, z.B. der Event-Service MSGTAC

Die Umstellung des Verfahrens (mit/ohne Drucksteuerung) wirkt in UTM-Cluster-Anwendungen Cluster-global.

Für die Drucksteuerung stellt openUTM ein besonderes Quittungsverfahren zur Verfügung. Um dieses Verfahren nutzen zu können, müssen Sie vom Automatikmodus in den Quittungsmodus umschalten. Wie sich Automatikmodus und Quittungsmodus voneinander unterscheiden, ist im Folgenden beschrieben.

Automatikmodus - Ausgaben ohne Drucksteuerung

Im Automatikmodus steuert openUTM die Ausgabe am Drucker. Die Ausgabe läuft dann wie folgt ab:

openUTM schickt die erste Nachricht in der Queue an den Drucker und erhält vom Drucker eine positive oder negative Abdruckquittung.

Empfängt openUTM die positive Abdruckquittung, dann löscht openUTM den Auftrag aus der Queue und überträgt die nächste Nachricht zum Drucker usw.

Erhält openUTM vom Drucker eine negative Abdruckquittung, dann erzeugt openUTM die Meldung K046. Diese Meldung ist standardmäßig keinem Meldungsziel zugeordnet. Sie können für die Meldung ein Meldungsziel definieren. Wie Sie das machen und welche Ziele Sie der Meldung zuordnen können, ist im openUTM-Handbuch „Meldungen, Test und Diagnose“ beschrieben.

Meldungsziel von K046 kann z.B. der Event-Service MSGTAC sein. Mit der MSGTAC-Routine, die Sie selbst erstellen müssen (siehe openUTM-Handbuch „Anwendungen programmieren mit KDCS“), können Sie dann auf die Fehlersituation reagieren. Der MSGTAC kann z.B. den Quittungsmodus einschalten (siehe auch Kapitel "Message Queues administrieren, Druckausgabe steuern").

Quittungsmodus - Ausgabe mit Drucksteuerung

Im Quittungsmodus muss die Druckausgabe über Teilprogramme mit PADM-Aufrufen gesteuert werden. Druckausgaben im Quittungsmodus laufen wie folgt ab:

openUTM übergibt eine Nachricht an den Drucker. Wenn die Nachricht mit einer positiven Abdruckquittung beendet wird, wartet openUTM auf eine Bestätigung, um eine Nachrichtenabschlussbehandlung durchzuführen. Der Benutzer/Client kann direkt am Druckersteuer-LTERM bestätigen, oder es wird automatisch bestätigt, z.B. durch die MSGTAC-Routine. Für die MSGTAC-Routine erzeugt openUTM bei einer positiven Abdruckquittung die Meldung K045.

Zur Bestätigung der Druckausgabe muss ein Vorgang mit PADM-Aufrufen gestartet werden, der openUTM darüber informiert, ob der Druck zu wiederholen ist oder ob die nächste Nachricht gedruckt werden soll.

Mit PADM AI können Sie Informationen über zu quittierende Druckausgaben anfordern. Ein Benutzer/Client am Druckersteuer-LTERM kann sich also über solche Nachrichten informieren, er kann aber auch mit Hilfe der MSGTAC-Routine informiert werden.

Im Quittungsmodus erzeugt openUTM beim Empfang einer positiven Abdruckquittung die Meldung K045. Dieser Meldung können Sie das Meldungsziel MSGTAC zuordnen, damit openUTM sie an die MSGTAC-Routine übergibt. Die MSGTAC-Routine kann dann das Druckersteuer-LTERM über die angeforderte Quittung informieren.

Fehler bei der Druckausgabe (negative Abdruckquittung) werden wie im Automatikmodus behandelt.

Funktionen zur Drucksteuerung

Funktionen zur Drucksteuerung bietet openUTM mit dem Aufruf PADM. Welche Funktion PADM in einzelnen durchführt, ist abhängig von der Operationsmodifikation, die Sie im Feld *kcom* des Parameterbereichs an openUTM übergeben. Folgende Operationsmodifikationen stehen zur Verfügung:

- PADM AC zum Einschalten des Quittungsmodus.
- PADM AT zum Ausschalten des Quittungsmodus. Es wird wieder der Automatikmodus eingestellt.
- PADM PR zum Wiederholen einer Druckausgabe. Die Drucker-Nachricht wird auf demselben Drucker erneut ausgegeben.
- PADM OK zum Bestätigen von Druckausgaben.
- PADM AI zum Anfordern einer Liste der zu quittierenden Druckausgaben mit Informationen.

13.3.2.1 Quittungsmodus ein- oder ausschalten - PADM AC/AT

Mit PADM AC können Sie für einen Drucker des Druckersteuer-LTERMs oder für alle Drucker eines Druckersteuer-LTERMs den Quittungsmodus einschalten. Im Quittungsmodus läuft die Drucksteuerung nicht mehr automatisch ab. openUTM löscht den zugehörigen Druckauftrag erst aus der Queue, wenn ein PADM OK-Aufruf für diesen Drucker abgesetzt wird.

Mit PADM AT wird der Quittungsmodus ausgeschaltet. Die Druckausgabe läuft wieder im Automatikmodus ab.

Das Umschalten des Quittiermodus' wirkt in UTM-Cluster-Anwendungen Cluster-global.

Soll PADM AT/AC auf einen bestimmten Drucker wirken, dann müssen Sie im Feld *kcrn* die Drucker-Id des Druckers angeben. Wenn der Aufruf für alle Drucker des Druckersteuer-LTERMs gelten soll, dann müssen Sie in *kcrn* Leerzeichen angeben.

Das Ein- und Ausschalten des Quittungsmodus wirkt über das Beenden der Anwendung hinaus.

Beim Ausschalten des Quittungsmodus ist zu beachten, dass eine Druckausgabe, die noch mit Quittungsmodus gestartet wurde, vor dem Ausschalten jedoch noch nicht bestätigt wurde, auch im Automatikmodus bestätigt werden muss. D.h. openUTM behandelt weitere Druckaufträge für den entsprechenden Drucker erst, wenn ein PADM OK abgesetzt wurde.

KDCPADM MODE im Abschnitt "MODE - Quittungsmodus eines Druckers ändern"

13.3.2.2 Druckausgabe bestätigen oder wiederholen - PADM OK/PR

Diese Funktion kann nur genutzt werden, wenn der Quittungsmodus eingeschaltet ist. Mit dem Aufruf PADM OK wird eine Druckausgabe bestätigt. openUTM löscht den zugehörigen Asynchron-Auftrag aus der Queue des Druckers und kann den nächsten Druckauftrag behandeln.

Mit PADM PR wird die Druckausgabe wiederholt, z.B. nach einem Probedruck. Der Druckauftrag wird nicht aus der Queue gelöscht. Er bleibt an der ersten Stelle in der Queue stehen und wird noch einmal bearbeitet.

KDCPADM PRINT im Abschnitt "PRINT - Druckausgabe bestätigen / wiederholen"

13.3.2.3 Informationen über zu quittierende Druckaufträge abfragen - PADM AI

PADM AI liefert Informationen über zu quittierende Druckaufträge. Ist kein Druckauftrag zu quittieren, dann gibt PADM AI nur Leerzeichen zurück.

Folgende Informationen liefert openUTM zu jedem Druckauftrag zurück:

- Drucker-Id
- Auftrags-Id des Asynchron-Auftrags
- Benutzerkennung des Auftraggebers
- Zeitpunkt der Auftragsstellung
- bei zeitgesteuerten Aufträgen die Zielzeit
- positiver und/oder negativer Quittungs-Auftrag

Wollen Sie die zu quittierenden Druckausgaben für alle Drucker des Druckersteuer-LTERMs abfragen, dann müssen Sie wie folgt vorgehen:
Beim ersten PADM AI/PI-Aufruf im Teilprogramm übergeben Sie statt einer Drucker-Id Leerzeichen im Feld *kcrn* des Parameterbereichs. openUTM liefert dann im Nachrichtenbereich die Drucker-Id des ersten Druckers (neben anderen Informationen) zurück. Im Feld *kcrmf* steht die Drucker-Id des nächsten Druckers. Den Inhalt des Feldes *kcrmf* übergeben Sie im Feld *kcrn* des nächsten PADM AI-Aufrufs, usw. Beim letzten Drucker übergibt openUTM im Feld *kcrmf* Leerzeichen.

KDCPADM INFORM, LIST=ACK im Abschnitt "INFORM - Informieren über Drucker eines Druckersteuer-LTERMs"

13.3.3 Behandlung von Fehlern bei der Druckausgabe

Fehler werden bei der Druckausgabe mit und ohne Drucksteuerung gleich behandelt. In diesem Abschnitt wird beschrieben, mit welchen UTM-Mitteln Sie auf Funktionsstörungen eines Druckers reagieren können.

Hardwarefehler
Bei Hardwarefehlern gibt es folgende Eingriffsmöglichkeiten:

- Das Terminal, das dem Druckersteuer-LTERM zugeordnet ist, ist defekt. Dann kann dem Druckersteuer-LTERM durch die Administration ein anderes Terminal zugeordnet werden, z.B. mit dem Administrationskommando KDCSWTCH.
- Ein Drucker ist defekt. Dann kann dem LTERM-Partner des Druckers und damit seiner Message Queue ein anderer Drucker zugeordnet werden, z.B. durch das KDCSWTCH-Kommando oder durch einen Vorgang mit PADM CA-Aufruf. Ist der LTERM-Partner einem Druckersteuer-LTERM zugeordnet, dann muss darauf geachtet werden, dass die Drucker-Id des „neuen“ Druckers im Bereich des Druckersteuer-LTERMs eindeutig ist.

Formatierungsfehler auf BS2000-Systemen
Treten bei der Umsetzung einer logischen Nachricht in eine physikalische (durch VTSU) oder eine formatierte Nachricht (durch FHS) Fehler auf, dann löscht openUTM die Nachricht und erzeugt einen Dump. Ist die Nachricht Hauptauftrag eines Auftrags-Komplexes, dann wird der negative Quittungs-Auftrag gestartet.

Fehlerbehandlung über MSGTAC-Routinen

Eine gezielte Fehlerbehandlung ist über den Event-Service MSGTAC möglich. Da das UTM-Teilprogramm administrationsberechtigt ist, kann es alle Drucker der Anwendung administrieren und die Drucksteuerung für alle Drucker durchführen.

openUTM erzeugt in Fehlerfällen die Meldung K046. Dieser Meldung können Sie das Meldungsziel MSGTAC zuordnen (siehe openUTM-Handbuch „Meldungen, Test und Diagnose“). Beim Auftreten dieser Meldung wird dann die MSGTAC-Routine durchlaufen. Die MSGTAC-Routine kann PADM-Aufrufe enthalten. Sie kann z.B.

- den Quittungsmodus einschalten und im weiteren die Druckausgaben vom Druckersteuer-LTERM bestätigen oder wiederholen lassen.
- den LTERM-Partner des Druckers, d.h. die Queue des Druckers, einem anderen Drucker zuordnen.
- einen bestimmten Benutzer/Client über den Fehler informieren.

openUTM erzeugt die Meldung K046 bei folgenden Fehlern:

- negative Abdruckquittung vom Drucker erhalten
- Wiederholung einer Druckausgabe
- Verbindungsaufbau zum Drucker nicht möglich

13.4 UTM-Teilprogramme für DADM- und PADM-Funktionen

Mit openUTM werden die KDCS-Teilprogramme KDCDADM und KDCPADM ausgeliefert, die Ihnen alle Leistungen der Aufrufe DADM und PADM zur Verfügung stellen, ohne dass Sie selbst Teilprogramme zur Administration von Message Queues und Druckern und zur Drucksteuerung erstellen müssen.

- KDCDADM stellt die Funktionen von DADM zur Administration von Nachrichten zur Verfügung.
- KDCPADM stellt die Funktionen von PADM zur Administration von Druckern und Steuerung der Ausgabe von Nachrichten an Druckern zur Verfügung.

Die für KDCDADM und KDCPADM benötigten ISP-Syntaxtabellen sind in KDCDAISP enthalten.

Vorgänge, in denen die Teilprogramme KDCDADM und KDCPADM ablaufen, laufen als Dialog-Transaktionen in einem Dialog-Schritt ab. KDCDADM und KDCPADM erwarten die Eingaben im Zeilenmodus, formatierte Eingaben werden abgewiesen. Die Ergebnisse werden von KDCDADM und KDCPADM ebenfalls im Zeilenmodus ausgegeben. KDCDADM und KDCPADM geben englische Meldungen aus.

KDCDADM, KDCPADM und KDCDAISP werden als übersetzte Objekte bzw. Objektmodule ausgeliefert. Damit Sie die Teilprogramme inklusive ISP-Syntaxbeschreibung nutzen können, müssen Sie sie zu Ihrem Anwendungsprogramm binden und die Teilprogramme sowie Transaktionscodes, über die die Teilprogramme gestartet werden können, in die Konfiguration der Anwendung aufnehmen.

In openUTM auf BS2000-Systemen sind die Objektmodule in der LMS-Bibliothek SYSLIB.UTM.070.EXAMPLE abgelegt.

In openUTM auf Unix- und Linux-Systemen finden Sie die Objekte in der Bibliothek `libsample` unter dem Pfad *utmpfad*`/sample/sys`.

In openUTM auf Windows-Systemen finden Sie die Objekte in der Bibliothek *utmpfad*`\sys\libwork.lib`.

13.4.1 KDCDADM und KDCPADM generieren

Die Teilprogramme KDCDADM und KDCPADM müssen entweder statisch mit KDCDEF oder dynamisch in die Konfiguration eingetragen werden. Damit Sie die Funktionen von KDCDADM und KDCPADM nutzen können, müssen Sie den Teilprogrammen Dialog-Transaktionscodes zuordnen. Der TAC-Name kann beliebig gewählt werden. Im folgenden Beispiel wird KDCDADM der Transaktionscode *tacdadm* und KDCPADM der Transaktionscode *tacpadm* zugeordnet.

Beispiel für die KDCDEF-Generierung

- *BS2000-Systeme*:

```
PROGRAM  KDCDADM,COMP=ILCS
PROGRAM  KDCPADM,COMP=ILCS
TAC      TACDADM,PROGRAM=KDCDADM,CALL=FIRST,TYPE=D
TAC      TACPADM,PROGRAM=KDCPADM,CALL=FIRST,TYPE=D
```

- *Unix-, Linux- und Windows-Systeme*:

```
PROGRAM  KDCDADM,COMP=C
PROGRAM  KDCPADM,COMP=C
TAC      tacdadm,PROGRAM=KDCDADM,CALL=FIRST,TYPE=D
TAC      tacpadm,PROGRAM=KDCPADM,CALL=FIRST,TYPE=D
```

Bei der KDCDEF-Generierung der Anwendung müssen Sie zusätzlich Folgendes beachten:
Die in MAX SPAB= angegebene Länge des Standard Primären Arbeitsbereichs muss groß genug sein, um den KDCS-Parameterbereich aufnehmen zu können.

13.4.2 KDCDADM - Nachrichten administrieren

Das Teilprogramm KDCDADM ermöglicht die Administration von Nachrichten in Message Queues. KDCDADM umfasst drei Funktionen. Jede dieser Funktionen rufen Sie auf, indem Sie den Transaktionscode, den Sie dem Teilprogramm KDCDADM zugeordnet haben (im Folgenden *tacdadm* genannt), zusammen mit einigen Operanden angeben. Welche Operanden das sind, ist in diesem Abschnitt beschrieben.

KDCDADM umfasst folgende Funktionen:

- Stornieren von Nachrichten, d.h. Löschen aus der Message Queue (DELETE)
- Informieren über Nachrichten in einer Message Queue (INFORM)
- Vorziehen einer Nachricht in der Message Queue (NEXT)
- Verschieben von Nachrichten aus der Dead Letter Queue (MOVE)

Geben Sie `tacdadm HELP` an, dann informiert openUTM über die Syntax der KDCDADM-Aufrufe. openUTM gibt eine kurze Beschreibung der Funktionen aus.

13.4.2.1 DELETE - Nachricht aus Message Queue löschen

Wenn Sie *tacdadm* zusammen mit dem Operanden DELETE angeben, dann können Sie Nachrichten aus einer Message Queue löschen.

Sie können:

- eine bestimmte Nachricht löschen.
  In diesem Fall müssen Sie die Message Queue und die Nachricht eindeutig identifizieren. Die Message Queue identifizieren Sie, je nach Typ, über den TAC-Namen, den Namen des LTERM-Partners, die Benutzerkennung oder den Namen der temporären Queue. Die Nachricht identifizieren Sie über ihre Auftrags-Id und den Zeitpunkt, an dem die Nachricht erzeugt wurde. Beides können Sie mit *tacdadm* INFORM ermitteln.
- alle Nachrichten, die zur Zeit in einer Message Queue zwischengespeichert sind, löschen.
  Damit werden alle Nachrichten gelöscht, die noch nicht vom Empfänger (TAC, LTERM-Partner, Benutzerkennung, temporäre Queue) bearbeitet werden.

```
tacdadm     DELETE
            ,DESTINATION=destination
            [ ,DEST-TYPE = { LTERM | TAC | USER | QUEUE } ]
            ,DPUTID={ ALL | dput-id,GENTIME=time [,CHAINMSG= {ACT | DEL} ] }
```

DELETE — eine Nachricht stornieren bzw. alle Nachrichten, die in einer Message Queue stehen, stornieren.

DESTINATION=destination

bezeichnet die Message Queue des Empfängers, die die zu stornierende Nachricht enthält. Für *destination* müssen Sie den Namen eines TACs, eines LTERM-Partners, einer Benutzerkennung oder den Namen einer temporären Queue angeben.

DEST-TYPE= — bezeichnet den Typ des Empfängers (*destination*). Mögliche Angaben sind:

- LTERM — Empfänger ist ein LTERM-Partner.
- TAC — Empfänger ist ein TAC oder eine TAC-Queue.
- USER — Empfänger ist die Queue einer Benutzerkennung.
- QUEUE — Empfänger ist eine temporäre Queue.

DPUTID= — In DPUTID geben Sie an, welche Nachricht storniert werden soll.

- ALL — Alle Nachrichten der Message Queue des Empfängers in *destination* sollen storniert werden.
- dputid — Es soll eine Nachricht der Queue storniert werden. Für *dputid* müssen Sie dann die Auftrags-Id der zu stornierenden Nachricht angeben.

GENTIME=time — müssen Sie nur angeben, wenn Sie eine bestimmte Nachricht aus einer Queue löschen wollen (bei DPUTID=*dputid*).
In diesem Fall müssen Sie für *time* den Zeitpunkt angeben, an dem die Nachricht erzeugt wurde.

time ist in der Form (`ddd,hh,mm,ss`) anzugeben, wobei *ddd* den Industrietag angibt, *hh* die Stunde, *mm* die Minuten und *ss* die Sekunden.
openUTM benötigt *time*, um die zu löschende Nachricht eindeutig identifizieren zu können.

CHAINMSG= gibt an, ob beim Löschen eines Auftrags-Komplexes (DPUT-Auftrag mit Quittungsaufträgen) der negative Quittungs-Auftrag aktiviert werden soll oder nicht.

ACT Der negative Quittungs-Auftrag wird aktiviert, falls er existiert.

DEL Der negative Quittungs-Auftrag wird ebenfalls gelöscht.

Standard: ACT

Ergebnis

openUTM schickt eine Meldung an den LTERM-Partner/LPAP-Partner, über den das Kommando aufgerufen wurde. An der Meldung können Sie erkennen, ob der Auftrag angenommen oder abgelehnt wurde. Ob openUTM den Auftrag erfolgreich durchführen konnte, müssen Sie durch ein folgendes KDCDADM INFORM abfragen.

13.4.2.2 INFORM - Über Message Queues und Nachrichten informieren

Mit *tacdadm* INFORM können Sie sich über Message Queues informieren. openUTM liefert zu den einzelnen Nachrichten in der Queue immer folgende Informationen zurück:

- die Auftrags-Id, die Sie z.B. beim Löschen einer Nachricht benötigen
- die Benutzerkennung, über die die Nachricht erzeugt wurde
- den Zeitpunkt, zu dem die Nachricht erzeugt wurde
- bei zeitgesteuerten Nachrichten den Zeitpunkt, ab dem die Nachricht bearbeitet werden soll
- die Information, ob zu der Nachricht ein positiver oder negativer Quittungs-Auftrag gehört.

Bei der ausführlichen Information (LIST=LONG) gibt openUTM zusätzlich die Benutzerinformationen aus, die mit DPUT NI geschrieben worden sind.

Die Listen mit den Informationen, die openUTM zurückliefert, können u.U. sehr umfangreich sein. Deshalb besteht die Möglichkeit:

- die Ausgabe auf einen Drucker umzulenken (OUT).
- die Ausgabe einzuschränken, indem Sie die Auftrags-Id der Nachricht angeben, mit der die Ausgabeliste beginnen soll. Die Listen sind nach Auftrags-Ids aufsteigend geordnet. Geben Sie in CONT eine Auftrags-Id an, dann beginnt die Liste mit dieser Nachricht. Die Informationen zu den Nachrichten, deren Auftrags-Id lexikalisch kleiner ist als die angegebene Auftrags-Id, werden nicht ausgegeben.

```
tacdadm     INFORM
           ,DESTINATION=destination
           [ ,CONT=dputid ]
           [ ,DEST-TYPE = { LTERM | TAC | USER | QUEUE } ]
           [ ,LIST={ SHORT| LONG } ]
           [ ,OUT={ KDCDISP | ltermname } ]
```

INFORM Übersichtsliste über die Nachrichten in einer Message Queue ausgeben lassen.

DESTINATION=destination

bezeichnet die Message Queue des Empfängers der Nachrichten, über die openUTM informieren soll. Für *destination* müssen Sie den Namen eines TACs, eines LTERM-Partners, einer Benutzerkennung oder den Namen einer temporären Queue angeben.

DEST-TYPE= bezeichnet den Typ des Empfängers (*destination*). Mögliche Angaben sind:

LTERM oder TAC

Empfänger ist ein TAC, eine TAC-Queue oder ein LTERM-Partner.

USER Empfänger ist die Queue einer Benutzerkennung.

QUEUE Empfänger ist eine temporäre Queue.

CONT=dputid

steuert den Umfang der Ausgabe. Für *dputid* können Sie die Auftrags-Id der Nachricht angeben, mit der die Liste der Informationen beginnen soll. Die Liste enthält nur Informationen zu Nachrichten, deren Auftrags-Id lexikalisch größer als *dputid* ist, und zu der Nachricht mit der Auftrags-Id *dputid*.

LIST= bestimmt den Umfang der Informationen, die openUTM ausgeben soll.

SHORT Die mit DPUT NI erzeugten Benutzerinformationen werden nicht mit ausgegeben.

LONG Die mit DPUT NI geschriebenen Benutzerinformationen werden mit ausgegeben.

Standard: SHORT

OUT= gibt an, wo openUTM die Informationen ausgeben soll.

KDCDISP openUTM gibt die Informationen an dem Terminal aus, an dem die Informationen angefordert wurden, bzw. openUTM übergibt die Informationen an den Client, der die Informationen angefordert hat.

ltermname openUTM gibt die Informationen auf einen Drucker aus. Für *ltermname* ist der Name des LTERM-Partners anzugeben, dem der Drucker zugeordnet ist.

Ergebnis

Bei LIST=SHORT

User-id	DPUT-id	Gen-time	Start-time	Pos/Neg	Dest.
user1	dput-id	time1	time2	p/n/p n	dest1

Dabei bedeuten:

User-id

Benutzerkennung oder “*NONE“, wenn der Benutzer gelöscht wurde, der die Nachricht erzeugt hat.

DPUT-id

Auftrags-Id der Nachricht.

Gen-time

Zeitpunkt, zu dem die Nachricht erzeugt wurde.
Die Zeit wird in der Form *ddd/hh:mm:ss* ausgegeben.

Dabei bedeuten:
ddd Industrietag, *hh* Stunden, *mm* Minuten, *ss* Sekunden.

Start-time

wird nur für zeitgesteuerte Nachrichten (DPUT-Nachrichten) ausgegeben. *Start-time* ist der früheste Zeitpunkt, ab dem der Auftrag bearbeitet werden soll. Das Ausgabeformat der Zeit ist dasselbe wie bei *Gen-time*.

Pos/Neg

gibt an, ob ein positiver bzw. negativer Quittungs-Auftrag existiert.
Das Anzeigefeld enthält ein „p“, wenn ein positiver Quittungs-Auftrag existiert, ein „n“, wenn ein negativer Quittungs-Auftrag existiert. „p n“ bedeutet, dass ein positiver und ein negativer Quittungs-Auftrag existieren.

Dest.

Empfänger der Nachricht. Für die Dead Letter Queue wird hier das ursprüngliche Ziel der Nachricht ausgegeben, also der Name eines Asynchron-TACs, einer TAC-Queue, eines LPAP-Partners oder eines OSI-LPAP-Partners. Andernfalls ist das Feld leer.

Bei LIST=LONG

Zusätzlich zu den Informationen, die bei LIST=SHORT ausgegeben werden, werden (in der nächsten Zeile) die ersten 79 Byte der Benutzerinformation (DPUT NI-Nachricht) ausgegeben. Es wird Folgendes ausgegeben:

```
       User-id  DPUT-id  Gen-time   Start-time   Pos/Neg    Dest.
       user     dput-id  time1      time2        p/n/p n    dest1
User info:
   xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

13.4.2.3 MOVE - Nachrichten aus der Dead Letter Queue verschieben

Die Dead Letter Queue besteht aus Nachrichten, die nicht verarbeitet werden konnten. Um diese Nachrichten evtl. nach einer Fehlerbehebung noch verarbeiten zu können, müssen sie entweder ihrem ursprünglichen Ziel oder einem neuen Ziel zugeordnet werden.

Mit *tacdadm* MOVE können Sie einzelne Nachrichten oder alle Nachrichten, die in der Dead Letter Queue gespeichert wurden, entsprechend verschieben. Die Nachrichten können der jeweiligen ursprünglichen Message Queue oder einem beliebigen neuen Ziel vom gleichen Typ (Asynchron-TAC/TAC-Queue, LPAP-Partner, OSI-LPAP-Partner) zugeordnet werden.

```
tacdadm     MOVE
           ,DESTINATION = { *ORIG | destination }
           ,DPUTID = { ALL | dputid, GENTIME = (ddd,hh,mm,ss) }
```

MOVE Nachrichten der Dead Letter Queue verschieben.

DESTINATION=

gibt das neue Ziel der Nachricht an.

*ORIG die Nachricht soll ihrem ursprünglichen Ziel zugeordnet werden. Verwenden Sie DESTINATION=*ORIG zusammen mit DPUTID=ALL, werden alle Nachrichten ihren jeweiligen ursprünglichen zielen zugeordnet.

destination

Name des neuen Ziels der Nachricht bzw. aller Nachrichten mit passendem ursprünglichen Ziel (Asynchron-TAC/TAC-Queue, LPAP-Partner, OSI-LPAP-Partner).

> ¡ Wenn Sie mehrere Nachrichten verschieben (DPUTID=ALL), dann verbleiben die Nachrichten, deren ursprüngliches Ziel nicht zum neuen Ziel passt, in der Dead Letter Queue.

DPUTID= Identifizierung der Nachricht, die verschoben werden soll.

ALL alle Nachrichten der Dead Letter Queue.

dputid Auftrags-Id der Nachricht.

GENTIME=(ddd,hh,mm,ss)

Zeitpunkt der Erzeugung der Nachricht. Dabei bedeuten:

ddd Industrietag, *hh* Stunden, *mm* Minuten, *ss* Sekunden.

Ergebnis

Der Auftrag zum Verschieben aller Nachrichten an die jeweiligen ursprünglichen Ziele wird ohne Fehlermeldung angenommen, wenn einzelne oder alle ursprünglichen Ziele nicht mehr existieren.

openUTM erzeugt eine Meldung, der Sie entnehmen können, ob der Auftrag angenommen wurde oder nicht. Die Meldung wird am Terminal des Auftraggebers ausgegeben.

Ob die Nachrichten tatsächlich verschoben wurden, müssen Sie mit eigenen KDCDADM-Kommandos überprüfen.

> **i** Mit openUTM wird das Beispielprogramm DADMMVS bzw. dadmmvsc für das selektive Verschieben von Nachrichten aus der Dead Letter Queue ausgeliefert. Das Dialogprogramm verschiebt alle Nachrichten der Dead Letter Queue mit vorgegebenem ursprünglichen Ziel an ein vorgegebenes neues Ziel. Die Beschreibung des Beispielprogramms finden Sie im jeweiligen System-spezifischen openUTM-Handbuch „Einsatz von UTM-Anwendungen“.

13.4.2.4 NEXT - Nachrichten in der Message Queue vorziehen

Mit *tacdadm* NEXT können Sie eine Nachricht, die an einer beliebigen Stelle in der Message Queue steht, an die erste Stelle vorziehen. D.h. diese Nachricht ist dann die nächste Nachricht, die von dem Empfänger bearbeitet wird.

Zeitgesteuerte Nachrichten (DPUT-Aufträge) können Sie nur vorziehen, wenn die angegebene Startzeit (der früheste Ausführungszeitpunkt) bereits erreicht ist.

Syntax:

```
tacdadm    NEXT
           ,DPUTID=dputid, GENTIME=(ddd,hh,mm,ss)
```

NEXT Die in *dputid* angegebene Nachricht wird an die erste Stelle der Message Queue gesetzt.

DPUTID=dputid

Auftrags-Id der Nachricht, die vorgezogen werden soll.

GENTIME=(ddd,hh,mm,ss)

Zeitpunkt, zu dem die Nachricht erzeugt wurde.
Dabei bedeuten:
ddd Industrietag, *hh* Stunden, *mm* Minuten, *ss* Sekunden.

Ergebnis

openUTM erzeugt eine Meldung, der Sie entnehmen können, ob der Auftrag angenommen wurde oder nicht. Die Meldung wird am Terminal des Auftraggebers ausgegeben bzw. an den Client, der den Auftrag gestartet hat.

13.4.3 KDCPADM - Drucksteuerung und Drucker-Administration

Das Teilprogramm KDCPADM ermöglicht die Administration von Druckern und die Steuerung von Druckausgaben. KDCPADM umfasst fünf Funktionen. Jede dieser Funktionen rufen Sie auf, indem Sie den Transaktionscode, den Sie dem Teilprogramm KDCPADM zugeordnet haben (im Folgenden *tacpadm* genannt), zusammen mit einigen Operanden angeben. Welche Operanden das sind, ist in diesem Abschnitt beschrieben.

KDCPADM umfasst folgende Funktionen zur Drucksteuerung:

- Druckausgabe bestätigen bzw. einen Ausdruck wiederholen lassen (PRINT)
- Umschalten zwischen Quittungsmodus und Automatikmodus (MODE)

KDCPADM umfasst folgende Funktionen zur Druckeradministration:

- Status eines Druckers ändern (STATE).
  Sie können einen Drucker sperren, eine existierende Sperre aufheben oder die Verbindung zu einem Drucker auf- bzw. abbauen.

  Das Sperren und Entsperren von Druckern wirkt in UTM-Cluster-Anwendungen Cluster-global.
- Einem LTERM-Partner, d.h. einer bestimmten Drucker-Queue, einen anderen bzw. einen weiteren Drucker zuordnen (SWITCH).
  So können Sie, z.B. bei einer Störung, Druckaufträge von einem anderen Drucker bearbeiten lassen, oder Druckerbündel erzeugen.

  Diese Funktion ist nur in stand-alone UTM-Anwendungen erlaubt.
- Informieren über die Drucker, die einem Druckersteuer-LTERM zugeordnet sind (INFORM).

Geben Sie `tacpadm HELP` an, dann informiert openUTM über die Syntax der KDCPADM-Aufrufe. openUTM gibt eine kurze Beschreibung der Funktionen aus.

13.4.3.1 INFORM - Informieren über Drucker eines Druckersteuer-LTERMs

Mit *tacpadm* INFORM können Sie sich über Drucker und die einem Drucker zugeordnete Message Queue informieren.

openUTM liefert folgende Informationen über die Drucker des Druckersteuer-LTERMs zurück:

- Name des LTERM-Partners, dem der Drucker zugeordnet ist.
- Status des Druckers, d.h. openUTM gibt an, ob der Drucker zur Zeit mit der Anwendung verbunden ist und ob der Drucker gesperrt ist.
- Quittungsmodus, d.h. openUTM gibt an, ob für den Drucker der Automatik- oder der Quittungsmodus eingestellt ist.
- Anzahl der Ausgabe-Aufträge, die zur Zeit in der Queue des Druckers bzw. des Druckerbündels zwischengespeichert sind.
- Anzahl der zeitgesteuerten Ausgabe-Aufträge, die zur Zeit in der Queue zwischengespeichert sind.

openUTM liefert folgende Informationen über die Ausgabe-Aufträge in der Queue eines Druckers oder Druckerbündels zurück:

- Zeitpunkt, zu dem der Auftrag erzeugt wurde.
- bei zeitgesteuerten Aufträgen den Zeitpunkt, ab dem der Auftrag bearbeitet werden soll.
- die Information, ob zu dem Auftrag ein positiver oder negativer Quittungs-Auftrag gehört.

Die Listen mit den Informationen, die openUTM zurückliefert, können u.U. sehr umfangreich sein. Deshalb besteht die Möglichkeit:

- die Ausgabe auf einen Drucker umzulenken (OUT).
- die Ausgabe einzuschränken, indem Sie die Auftrags-Id des Auftrags angeben, mit dem die Ausgabeliste beginnen soll. Die Listen sind nach Auftrags-Ids aufsteigend geordnet. Geben Sie in CONT eine Auftrags-Id an, dann beginnt die Liste mit diesem Auftrag. Die Informationen zu den Aufträgen, deren Auftrags-Id kleiner ist als die angegebene Auftrags-Id, werden nicht ausgegeben.

Syntax:

```
tacpadm     INFORM
           ,LIST= { PRINTERS | ACK }
           [ ,CID=cid1 ]
           [ ,CONT=cid2 ]
           [ ,OUT={ KDCDISP | ltermname1 } ]
           [ ,CTERM=ltermname ]
```

INFORM	Übersichtsliste über Drucker oder die Ausgabe-Aufträge ausgeben.
CID=cid1	(**C**ontrol-**ID**) Drucker-Id des Druckers. Geben Sie *cid* nicht an, dann liefert openUTM Information zu allen Druckern bzw. Message Queues zurück, die dem Druckersteuer-LTERM zugeordnet sind.

LIST= gibt an, welche Informationen angefordert werden

PRINTERS

Informationen über Drucker

ACK Informationen über die Ausgabe-Aufträge in den Queues der Drucker, die noch quittiert werden müssen.

CONT=cid2 steuert den Umfang der Ausgabe. Für *cid2* können Sie die Auftrags-Id des Druckers angeben, mit dem die Liste der Informationen beginnen soll.

Die Angabe von CONT= ist nur sinnvoll, wenn die Ausgabe zu einem vorherigen INFORM-Aufruf nicht auf eine Bildschirmseite passte. Sie können, um die Ausgabe fortzusetzen, bei dem Folgeaufruf in *cid2* die Drucker-Id des letzten Druckers der vorherigen Ausgabe angeben.

OUT= gibt an, wo openUTM die Informationen ausgeben soll.

KDCDISP openUTM gibt die Informationen an dem Terminal aus, an dem die Informationen angefordert wurden bzw. openUTM übergibt die Informationen an den Client, der die Informationen angefordert hat.

ltermname1

openUTM gibt die Informationen auf dem Drucker aus. Für *ltermname1* ist der Name des LTERM-Partners anzugeben, dem der Drucker zugeordnet ist.

Standard: KDCDISP

CTERM=ltermname

Druckersteuer-LTERM, zu dem der Drucker *cid1* gehört. Für *ltermname* ist der Name des Druckersteuer-LTERMs anzugeben. Wird das Kommando nicht am Druckersteuer-LTERM des Druckers *cid1* eingegeben, dann muss der Benutzer, der das Kommando eingibt, administrationsberechtigt sein.

Standard:
Name des LTERM-Partners, an dem das Kommando eingegeben wird.

Ergebnis

Bei LIST=PRINTERS

```
Control-id  State  Connected  Mode      LTERM-name # of msg.:  output  delayed
cid1        Y/N    Y/N        auto/ack  lterm1                 num1    num2
```

Control-id

Drucker-Id des Druckers

State gibt an, ob der Drucker gesperrt ist (N) oder nicht (Y).

Connected

gibt an, ob die Verbindung zum Drucker aufgebaut ist (Y) oder nicht (N).

Mode gibt an, ob der Quittungsmodus (ack) oder der Automatikmodus (auto) eingestellt ist.

LTERM-name

Name des LTERM-Partners, der dem Drucker zugeordnet ist.

output Anzahl der Ausgabe-Aufträge, die derzeit in der Queue des Druckers zwischengespeichert sind.

delayed

Anzahl der zeitgesteuerten Ausgabe-Aufträge (DPUT-Aufträge), die derzeit zur Bearbeitung in der Queue des Druckers zwischengespeichert sind, und deren Startzeitpunkt noch nicht erreicht ist.

Bei LIST=ACK

```
Control-id  User-id  DPUT-id  Gen-time  Start-time  Pos/Neg chain msg
cid1        user1    dput-id  time1     time2       p / n / p n
```

Dabei bedeuten:

Control-id

Drucker-Id des Druckers

User-id

Benutzerkennung oder “*NONE“, wenn der Benutzer gelöscht wurde, der den Auftrag erzeugt hat.

DPUT-id

Auftrags-Id des Asynchron-Auftrags

Gen-time

Zeitpunkt, zu dem der Ausgabe-Auftrag erzeugt wurde.
Die Zeit wird in der Form `ddd/hh:mm:ss` ausgegeben. Dabei bedeuten:*ddd* Industrietag, *hh* Stunden, *mm* Minuten, *ss* Sekunden.

Start-time

wird nur für zeitgesteuerte Aufträge (DPUT-Aufträge) ausgegeben.
Start-time ist der früheste Zeitpunkt, ab dem der Auftrag bearbeitet werden soll. Das Ausgabeformat der Zeit ist dasselbe wie bei *Gen-time*.

Pos/Neg chain msg.

gibt an, ob ein positiver bzw. negativer Quittungs-Auftrag existiert.
Das Anzeigefeld enthält ein „p“, wenn ein positiver Quittungs-Auftrag existiert, ein „n“, wenn eine negative Folgenachricht existiert. „p n“ bedeutet, dass ein positiver und ein negativer Quittungs-Auftrag existieren.

13.4.3.2 MODE - Quittungsmodus eines Druckers ändern

Mit *tacpadm* MODE können Sie den Quittiermodus ändern. Sie können vom Automatikmodus auf den Quittungsmodus umschalten und umgekehrt.

Das Ändern des Quittiermodus' wirkt in UTM-Cluster-Anwendungen Cluster-global.

Syntax:

```
tacpadm    MODE
           [ ,CID=cid ]
             ,ACT={ ACK | AUTO}
           [ ,CTERM=ltermname ]
```

MODE — Umschalten zwischen Automatikmodus und Quittungsmodus eines Druckers.

CID=cid — (**C**ontrol-**ID**)
Drucker-Id des Druckers, der administriert werden soll. Geben Sie *cid* nicht an, dann bezieht sich der Aufruf auf alle Drucker, die dem Druckersteuer-LTERM *ltermname* zugeordnet sind.
Wird der Aufruf nicht am Druckersteuer-LTERM abgesetzt, dann muss der Benutzer administrationsberechtigt sein.

ACT= — Aktion, die durchgeführt werden soll, Pflichtoperand.

- ACK — Umschalten auf den Quittungsmodus, d.h. jede Druckausgabe muss bestätigt werden (z.B. mit PRINT,...,ACT=NEXT).
- AUTO — Automatikmodus einschalten, d.h. die Druckausgabe muss nicht quittiert werden. Falls für *cid* eine Drucker-Id angegeben wurde, dann wird die letzte Druckausgabe für diesen Drucker automatisch bestätigt.

CTERM=ltermname

Druckersteuer-LTERM, zu dem der Drucker *cid* gehört. Für *ltermname* ist der Name des Druckersteuer-LTERMs anzugeben. Wird das Kommando nicht an diesem Druckersteuer-LTERM eingegeben, dann muss der Benutzer, der den Vorgang startet, administrationsberechtigt sein.

Standard:
Name des LTERM-Partners, an dem das Kommando eingegeben wird.

Ergebnis

openUTM liefert eine Meldung zurück, der Sie entnehmen können, ob der Auftrag angenommen oder abgelehnt wurde.

13.4.3.3 PRINT - Druckausgabe bestätigen / wiederholen

Mit *tacpadm* PRINT können Sie eine Druckausgabe bestätigen und die Bearbeitung des nächsten Auftrags veranlassen oder eine Druckausgabe wiederholen lassen. Voraussetzung für den Aufruf von *tacpadm* PRINT ist, dass zuvor der Quittungsmodus eingestellt wurde.

```
tacpadm     PRINT
           ,CID=cid
           [ ,ACT={ NEXT| REPEAT} ]
           [ ,CTERM=ltermname ]
```

PRINT — Druckausgabe bestätigen oder wiederholen.

CID=cid — (**C**ontrol-**ID**)
Drucker-Id des Druckers, auf den sich der Aufruf bezieht.

ACT= — Aktion, die ausgeführt werden soll.

- NEXT — Druckausgabe wird bestätigt und die Bearbeitung des folgenden Ausgabe-Auftrags veranlasst.
- REPEAT — Druckausgabe soll wiederholt werden.

Standard: NEXT

CTERM=ltermname

Name des Druckersteuer-LTERMs, zu dem der Drucker gehört. Wird das Kommando nicht an diesem Druckersteuer-LTERM eingegeben, dann muss der Benutzer, der den Vorgang startet, administrationsberechtigt sein.

Standard:
Name des LTERM-Partners, an dem das Kommando eingegeben wird.

Ergebnis

openUTM liefert eine Meldung zurück, der Sie entnehmen können, ob der Auftrag angenommen oder abgelehnt wurde.

13.4.3.4 STATE - Status eines Druckers ändern

Mit *tacpadm* STATE können Sie den Status eines Druckers ändern. Sie können:

- einen Drucker sperren bzw. einen zuvor gesperrten Drucker wieder freigeben.
- die Verbindung zu einem Drucker auf- bzw. abbauen.

```
tacpadm     STATE
           ,CID=cid
           ,ACT={ ON | OFF | CON | DIS | DISOFF }
           [ ,CTERM=ltermname ]
```

STATE — Status eines Druckers ändern.

CID=cid — (**C**ontrol-**ID**)

Drucker-Id des Druckers, dessen Status geändert werden soll.

ACT= — Aktion, die durchgeführt werden soll, Pflichtoperand.

- ON — einen gesperrten Drucker wieder zulassen.
- OFF — einen Drucker sperren, d.h. zu diesem Drucker kann keine Verbindung mehr aufgebaut werden. Ist der Drucker z.Zt. konnektiert, dann wird diese Verbindung abgebaut.

  On und OFF wirken in UTM-Cluster-Anwendungen Cluster-global.
- CON — die Verbindung zum Drucker aufbauen
- DIS — die Verbindung zum Drucker abbauen
- DISOFF — Verbindung zum Drucker abbauen und Drucker sperren

CTERM=ltermname

Name des Druckersteuer-LTERMs, zu dem der Drucker gehört. Wird das Kommando nicht an diesem Druckersteuer-LTERM eingegeben, dann muss der Benutzer, der den Vorgang startet, administrationsberechtigt sein.

Standard:
Name des LTERM-Partners, an dem das Kommando eingegeben wird.

Ergebnis

openUTM liefert eine Meldung zurück, der Sie entnehmen können, ob der Auftrag angenommen oder abgelehnt wurde.

13.4.3.5 SWITCH - Zuordnung Drucker zu LTERM-Partner ändern

Mit *tacpadm* SWITCH können Sie die Zuordnung zwischen LTERM-Partner und Drucker ändern. Diese Funktion ist nur in stand-alone UTM-Anwendungen erlaubt. Sie können:

- den LTERM-Partner dieses Druckers zusammen mit der zugehörigen Message Queue einem anderen Drucker zuordnen, der dann die in der Queue stehenden Druckaufträge bearbeitet. Damit können Sie z.B. bei Ausfall eines Druckers die an ihn gerichteten Ausgabe-Aufträge an einem anderen Drucker ausgeben lassen.
- Drucker zu Druckerbündeln zusammenfassen. Dabei ordnen Sie einem LTERM-Partner mehrere Drucker zu. Die Message Queue des LTERM-Partners wird dann von allen Druckern des Bündels gemeinsam abgearbeitet. Zu Druckerbündeln siehe auch openUTM-Handbuch „Anwendungen generieren“.

```
tacpadm     SWITCH
           ,CID=cid
           ,LTERM=ltermname1
           [ ,CTERM=ltermname ]
```

SWITCH Zuordnung zwischen Drucker und LTERM-Partner ändern

CID=cid (**C**ontrol-**ID**) Drucker-Id des Druckers, der einem anderen LTERM-Partner zugeordnet werden soll.

LTERM=ltermname1

Name des LTERM-Partners, dem der Drucker zugeordnet werden soll. Für *ltermname1* dürfen Sie nur einen LTERM-Partner angeben, der explizit für Drucker und andere Ausgabemedien generiert wurde. Ist dem LTERM-Partner bereits ein Drucker zugeordnet, dann wird diese Zuordnung nicht aufgelöst. Die Drucker werden zu einem Druckerbündel zusammengefasst.

CTERM=ltermname

Name des Druckersteuer-LTERMs, zu dem der Drucker *cid* gehört. Wird das Kommando nicht an diesem Druckersteuer-LTERM eingegeben, dann muss der Benutzer, der den Vorgang startet, administrationsberechtigt sein.

Standard:
Name des LTERM-Partners, an dem das Kommando eingegeben wird.

Ergebnis

openUTM liefert eine Meldung zurück, der Sie entnehmen können, ob der Auftrag angenommen oder abgelehnt wurde.

14 Anhang

- Programmschnittstelle zur Administration in COBOL
  - COPY-Elemente für die Programmschnittstelle in COBOL
  - KDCADMI-Funktionsaufruf
  - Hinweise zur Programmierung
- Beispielprogramme
  - Das C-Teilprogramm HNDLUSR (BS2000-Systeme)
  - Das C-Teilprogramm SUSRMAX
  - Das COBOL-Teilprogramm COBUSER
  - Das C-Teilprogramm ENCRADM
  - Das C-Teilprogramm ADJTCLT
- CALLUTM - Tool für Administration und Client-Server-Kommunikation (BS2000-Systeme)
  - Generierung
  - Beschreibung der CALLUTM-Programm-Anweisungen
  - Bestandteile, Systemumgebung, Softwarekonfiguration auf dem BS2000-System
  - Integration in eine UTM-Anwendung auf dem BS2000-System
  - Programmüberwachende Jobvariable auf dem BS2000-System
  - Meldungen von CALLUTM (BS2000-Systeme)

14.1 Programmschnittstelle zur Administration in COBOL

Die COBOL-Programmschnittstelle zur Administration entspricht weitgehend der in Kapitel "Programmschnittstelle zur Administration - KDCADMI" beschriebenen C/C++-Programmschnittstelle. Aus diesem Grund können Sie die Beschreibung der Programmschnittstelle in Kapitel "Programmschnittstelle zur Administration - KDCADMI" und die Ausführungen zum Funktionsumfang, zum Aufbau eigener Administrationsprogramme und zur zentralen und automatischen Administration (Kapitel "Objekte administrieren, Parameter einstellen", "Konfiguration dynamisch ändern", "Generierungsanweisungen aus der KDCFILE erzeugen", "Erstellen eigener Administrationsprogramme", "Zentrale Administration mehrerer Anwendungen", "Administration automatisieren" und "Zugriffsrechte und Zugriffsschutz") bei der Erstellung von eigenen Administrationsprogrammen in COBOL zu Rate ziehen. In diesem Abschnitt werden die Unterschiede aufgelistet, die Sie bei der Erstellung von COBOL-Administrationsprogrammen beachten müssen.

Folgende Unterschiede gegenüber der C/C++-Programmschnittstelle gibt es:

- Statt einer Include-Datei (*kcadminc.h*) mit allen Datenstrukturen werden für COBOL einzelne COPY-Elementen ausgeliefert. Dabei enthält jedes COPY-Element i.A. nur eine Datenstruktur (siehe Tabelle im Abschnitt „COPY-Elemente für die Programmschnittstelle in COBOL"). Damit haben Sie die Möglichkeit nur einzelne Datenstrukturen in die Programme zu kopieren, was das Programmieren u.U. erheblich erleichtert, z.B. beim Aufbau von Tabellen bei der Ein- und Ausgabe.
- Entsprechend den COBOL Konventionen werden in den Feldnamen Großbuchstaben statt Kleinbuchstaben und Bindestriche (-) statt Unterstriche (_; Underscore) verwendet.

  Beispiel: Dem C-Datenfeld *obj_type* entspricht in COBOL der Feldname OBJ-TYPE.

14.1.1 COPY-Elemente für die Programmschnittstelle in COBOL

Die Namen der COPY-Elemente für die Programmschnittstelle zur Administration beginnen alle mit dem Präfix KCA.

Die folgende Tabelle enthält die Namen der C-Datenstrukturen in alphabetischer Reihenfolge und gibt an, welches COPY-Element der C-Datenstruktur entspricht, bzw. welches COPY-Element bestimmte Definitionen enthält:

C-Datenstruktur / Definitionen	COBOL COPY-Element
Operationscodes und Sub-Operationscodes von KDCADMI (Werte von *opcode* und *subopcode1/2)*, die Objekttypen (Werte von *obj_type*) sowie Main- und Subcodes der Returncodes (Werte von *retcode*)	KCAOPRTC
Abdruckbare Strings für die Main- und Subcodes der Returncodes	KCAPRINC
kc_abstract_syntax_str	KCAABSTC
kc_access_point_str	KCAACCPC
kc_adm_parameter (Parameterbereich) und kc_id_area (Identifikationsbereich)	KCAPAIDC
kc_application_context_str	KCAAPLCC
kc_bcamappl_str	KCABCAMC
kc_change_application_str	KCAAPPLC
kc_character_set_str	KCACSETC
kc_cluster_curr_par_str	KCACCURC
kc_cluster_node_str	KCACLNOC
kc_cluster_par_str	KCACLPAC
kc_con_str	KCACONC
kc_create_statements_str	KCACREAC
kc_curr_par_str	KCACURRC
kc_db_info_str	KCADBIC
kc_diag_and_account_par_str	KCADACCC
kc_dyn_par_str	KCADYNC
kc_edit_str (nur auf BS2000-Systemen)	KCAEDITC
kc_encrypt_str	KCAENCRC
kc_encrypt_advanced_str	KCAENCAC
kc_gssb_str	[1]
kc_http_descriptor_str	KCAHTPDC
kc_kset_str	KCAKSETC

C-Datenstruktur / Definitionen	COBOL COPY-Element
kc_load_module_str	KCALMODC
kc_lock_mgtm_str	KCACLLKC
kc_lpap_str	KCALPAPC
kc_lses_str	KCALSESC
kc_lterm_str	KCALTRMC
kc_ltac_str	KCALTACC
kc_max_par_str	KCAMAXC
kc_msg_dest_par_str	KCAMSGDC
kc_message_module_str	KCAMSGMC
kc_mux_str (nur auf BS2000-Systemen)	KCAMUXC
kc_online_import_str	KCACLIMC
kc_osi_association_str	KCAOASSC
kc_osi_con_str	KCAOCONC
kc_osi_lpap_str	KCAOLPAC
kc_pagepool_str	KCAPGPLC
kc_ptc_str	KCAPTCC
kc_program_str	KCAPROGC
kc_pterm_str	KCAPTRMC
kc_queue_par_str	KCAQUPAC
kc_queue_str	KCAQUEUC
kc_sfunc_str	KCASFUNC
kc_shutdown_str	KCASHUTC
kc_signon_str	KCASIGNC
kc_subnet_str (nur auf Unix-, Linux- und Windows-Systemen)	KCASBNTC
kc_syslog_str	KCASLOGC
kc_system_par_str	KCASYSTC

C-Datenstruktur / Definitionen	COBOL COPY-Element
kc_tac_str	KCATACC
kc_tacclass_str	KCATCLC
kc_tasks_par_str	KCATASKC
kc_timer_par_str	KCATIMEC
kc_tpool_str	KCATPLC
kc_transfer_syntax_str	KCATRANC
kc_user_dyn1_str	KCAUSD1C
kc_user_dyn2_str	KCAUSD2C
kc_user_fix_str	KCAUSFXC
kc_user_str	KCAUSERC
kc_utmd_par_str	KCAUTMDC

1 Ein entsprechendes COPY-Element gibt es nicht, da kc_gssb_str nur aus dem 8 Zeichen langen Feld zur Aufnahme des GSSB-Namens (GS-NAME) besteht.

Die COPY-Elemente der COBOL-Programmschnittstelle finden Sie in folgenden Bibliotheken:

- für openUTM auf BS2000-Systemen in der Bibliothek SYSLIB.UTM.070.COB
- für openUTM auf Unix- und Linux-Systemen in dem Dateiverzeichnis *utmpfad* `/copy-cobol85` (Micro Focus Cobol-Compiler) bzw. *utmpfad* `/netcobol` (Compiler NETCOBOL von Fujitsu)
- für openUTM auf Windows-Systemen in dem Dateiverzeichnis *utmpfad* `\copy-cobol85` (Micro Focus Cobol-Compiler)

14.1.2 KDCADMI-Funktionsaufruf

Beim Aufruf von KDCADMI übergeben Sie wie an der C/C++-Schnittstelle vier Parameter an openUTM: den Parameterbereich KC-ADM-PARAMETER, den Identifikationsbereich ID-AREA, den Selektionsbereich SELECT-AREA und den Datenbereich DATA-AREA. Wie diese Bereiche zu versorgen sind, können Sie der Beschreibung der C/C++-Schnittstelle in Kapitel "Programmschnittstelle zur Administration - KDCADMI" entnehmen. Der Aufruf von KDCADMI muss folgendermaßen aussehen:

```
CALL "KDCADMI" USING KC-ADM-PARAMETER,
                     ID-AREA,
                     SELECT-AREA,
                     DATA-AREA.
```

14.1.3 Hinweise zur Programmierung

Bei der Erstellung von Administrationsprogrammen in COBOL müssen Sie folgende Punkte beachten:

- Wenn Sie mit den Tabellen der abdruckbaren Strings für die Returncodes arbeiten (COPY-Element KCAPRINC), dann müssen Sie beachten, dass in COBOL eine Tabelle mit dem Index „1“ beginnt, während die Werte der Returncodes bei „0“ anfangen. Den Zugriff auf einen abdruckbaren Returncode in der Tabelle können Sie z.B. folgendermaßen programmieren.

  ```
  MOVE MC-TEXT (KC-MAINCODE + 1) TO MC.
  MOVE SC-TEXT (KC-SUBCODE + 1) TO SC.
  ```

- Die Datenstruktur KC-ADM-PARAMETER mit den Aufrufparametern der Administrationsschnittstelle beginnt auf Stufe 1. KC-ADM-PARAMETER muss somit in die WORKING-STORAGE-SECTION oder die LOCAL-STORAGE-SECTION gelegt werden.
- Wenn eine Datenstruktur Unterstrukturen enthält, sollten diese generell vollqualifiziert angesprochen werden.

  Beispiel

  ```
  MOVE SIGN-YEAR IN SIGN-TIME-DATE IN KC-USER-STR TO ...
  ```

14.2 Beispielprogramme

Mit dem Produkt openUTM werden Beispielprogramme als Sourcecode und Objektmodule ausgeliefert, die Sie als Programmiervorlage für eigene Administrationsprogramme verwenden und nach Bedarf modifizieren, übersetzen und in Ihre Anwendung einbinden können. Bei den Beispielprogrammen handelt es sich um die Programme HNDLUSR (nur BS2000-Systeme), DADMMVS, PUBSUBA/PUBSUBD, SUSRMAX, COBUSER und ENCRADM (zur Beschreibung von DADMMVS und PUBSUBA/PUBSUBD siehe das jeweilige openUTM-Handbuch „Einsatz von UTM-Anwendungen“).

Auf BS2000-Systemen finden Sie die Sourcen und Objektmodule der Beispielprogramme, des Unterprogramms ERRCHCK und der Maske D0USER (IFG-Format) in der Bibliothek SYSLIB.UTM.070.EXAMPLE.

Auf Unix- und Linux-Systemen finden Sie den COBOL-Modul `COBUSER.cbl` im Verzeichnis *utmpfad*`/sample/src/mfcobol` bzw. .../`netcobol`. Die C-Beispielprogramme und das Unterprogramm ERRCHCK sind Bestandteil der Beispiel-Anwendung; sie sind nach Installation der Beispiel-Anwendung im zugehörigen Unterverzeichnis `utmsample/utm-c` zu finden.

Auf Windows-Systemen finden Sie den COBOL-Modul `COBUSER.cbl` im Verzeichnis *utmpfad*\`sample\src\mfcobol`. Die C-Beispielprogramme und das Unterprogramm ERRCHCK sind Bestandteil des Quick Start Kit; sie sind nach Installation des Quick Start Kit im zugehörigen Unterverzeichnis `\utmsample\utm-c` zu finden.

> Die Generierungsanweisungen für die C-Beispielprogramme sind bereits in den KDCDEF-Input-Dateien der Beispielanwendung (Unix- und Linux-Systeme) bzw. des Quick Start Kit (Windows-Systeme) eingetragen.

14.2.1 Das C-Teilprogramm HNDLUSR (BS2000-Systeme)

Mit HNDLUSR können Sie formatgesteuert folgende Aktionen durchführen:

- Eigenschaften von Benutzerkennungen abfragen und ändern,
- neue Benutzerkennungen in die Konfiguration eintragen,
- Benutzerkennungen aus der Konfiguration löschen.

Hinweise zur Generierung:

Das Teilprogramm muss im KDCDEF-Lauf folgendermaßen definiert werden:

```
PROGRAM HNDLUSR,COMP=ILCS
```

```
TAC HNDLUSR,PROGRAM=HNDLUSR,ADMIN=YES
```

```
FORMSYS ENTRY=KDCFHS,TYPE=FHS,LIB= Bibliothek mit Anschlussmodul zum Formatierungssystem
```

Das Teilprogramm verwendet intern die C-Routine ERRCHCK und das FHS-Format D0USER.

Hinweis zum Binden:

Das Teilprogramm HNDLUSR kann über eine RESOLVE-BY-AUTO-Anweisung in das Anwendungsprogramm eingebunden werden. Dadurch wird auch die Routine ERRCHCK implizit mit eingebunden.

Hinweis zum Starten:

Die Start-Prozedur für die Anwendung muss um die Parameter für das Formatierungssystem FHS erweitert werden:

```
.FHS MAPLIB=Formatbibliothek
.FHS ISTD=RUNP
```

Das FHS-Format D0USER müssen Sie vor dem Start der Anwendung aus der EXAMPLE-Bibliothek in die von Ihnen verwendete Formatbibliothek kopieren.

14.2.2 Das C-Teilprogramm SUSRMAX

Mit SUSRMAX können Sie folgende Aktionen durchführen:

- alle derzeit konnektierten Benutzerkennungen anzeigen,
- alle Benutzerkennungen anzeigen, die sich derzeit in einem Vorgang befinden,
- die aktuell eingestellten Werte für Anwendungsparameter anzeigen, die bei der KDCDEF-Generierung in MAX definiert werden und durch die Administration modifiziert werden können,
- diese Anwendungsparameter modifizieren.

Hinweise zur Generierung:

Das Teilprogramm muss im KDCDEF-Lauf wie folgt definiert werden:

- BS2000-Systeme:

  ```
  PROGRAM SUSRMAX,COMP=ILCS
  TAC SUSRMAX,PROGRAM=SUSRMAX,ADMIN=YES
  ```

- Unix, Linux- und Windows-Systeme:

  ```
  PROGRAM SUSRMAX,COMP=C
  TAC SUSRMAX,PROGRAM=SUSRMAX,ADMIN=YES
  ```

Das Teilprogramm benötigt für KB und SPAB die Mindestgrößen:

- 168 Bytes für KB-Programm Bereich und
- 6296 Bytes für den SPAB-Bereich.

Das Teilprogramm verwendet intern die C-Routine ERRCHCK.

Hinweis zum Binden:

Auf BS2000-Systemen kann das Teilprogramm SUSRMAX über eine RESOLVE-BY-AUTO-Anweisung in das Anwendungsprogramm eingebunden werden. Dadurch wird auch die Routine ERRCHCK implizit mit eingebunden.

Auf Unix-, Linux- und Windows-Systemen wird das Teilprogramm SUSRMAX automatisch in die Beispiel-Anwendung eingebunden.

14.2.3 Das COBOL-Teilprogramm COBUSER

Das Programm liest Informationen über angemeldete Benutzer und LTERM-Partner.

Hinweise zur Generierung:

Das Teilprogramm muss im KDCDEF-Lauf wie folgt definiert werden:

- BS2000-Systeme:

  ```
  PROGRAM COBUSER, COMP=ILCS
  TAC COBUSER, PROGRAM=COBUSER, ADMIN=YES
  ```

- Unix- und Linux-Systeme:

  ```
  PROGRAM COBUSER, COMP=COB2 (Micro Focus Compiler)
  PROGRAM COBUSER, COMP=NETCOBOL (NETCOBOL Compiler von Fujitsu)
  TAC COBUSER, PROGRAM=COBUSER, ADMIN=YES
  ```

- `Windows-Systeme:`

  ```
  PROGRAM COBUSER, COMP=COB2 (Micro Focus Compiler)
  TAC COBUSER, PROGRAM=COBUSER, ADMIN=YES
  ```

Hinweis zum Binden:

Auf BS2000-Systemen kann COBUSER über eine RESOLVE-BY-AUTO-Anweisung in das Anwendungsprogramm eingebunden werden.

Auf Unix- und Linux-Systemen müssen Sie die Bibliothek `libsample` einbinden, die unter dem Pfad *utmpfad* `/sample/sys` zu finden ist.

Auf Windows-Systemen muss das Objekt `COBUSER.obj` explizit eingebunden werden.

14.2.4 Das C-Teilprogramm ENCRADM

Mit dem Teilprogramm ENCRADM können die folgenden Funktionen zur Administration der Encryption-Software ausgeführt werden:

- neue RSA-Schlüsselpaare generieren
- neu erzeugte Schlüsselpaare aktivieren
- aktive und neu erzeugte Schlüsselpaare löschen
- öffentliche Schlüssel in Datei auslesen (sowohl aktive als auch neu erzeugte Schlüsselpaare)

ENCRADM ist auf Unix-, Linux- und Windows-Systemen Bestandteil der Beispiel-Anwendung.

Hinweise zur Generierung:

Das Teilprogramm muss mit folgenden KDCDEF-Anweisungen definiert werden:

- BS2000-Systeme:

```
PROGRAM ENCRADM, COMP=ILCS
TAC ENCRADM,PROGRAM=ENCRADM,ADMIN=YES
```

- Unix-, Linux und Windows-Systeme:

```
PROGRAM ENCRADM,COMP=C
TAC ENCRADM,PROGRAM=ENCRADM,ADMIN=YES
```

Das Teilprogramm benötigt für KB und SPAB als Minimalgrößen 200 Byte für KB-Programm Bereich und 4 KB für den SPAB.

Das Teilprogramm verwendet intern die C-Routine ERRCHCK.

Hinweis zum Binden:

Auf BS2000-Systemen wird das Teilprogramm ENCRADM über eine RESOLVE-BY-AUTO-Anweisung auf die EXAMPLE-Bibliothek in das Anwendungsprogramm eingebunden. Dadurch wird auch die Routine ERRCHCK implizit mit eingebunden.

Auf Unix-, Linux- und Windows-Systemen wird das Teilprogramm ENCRADM automatisch in die Beispiel-Anwendung eingebunden.

14.2.5 Das C-Teilprogramm ADJTCLT

Mit dem C-Teilprogramm ADJTCLT (adjust tac class) kann der Anwender steuern, wie die Prozesse (Tasks) auf die TAC-Klassen aufgeteilt werden, und zwar abhängig von der aktuellen Anzahl aller Prozesse und der aktuellen Anzahl der Asynchron-Prozesse. Dazu erstellt der Anwender eine Tabelle mit den gewünschten Einstellungen, siehe Abschnitt „TAC-Klassen-Tabelle erstellen“.

Das Programm wird als vollständiges Dialog- und Asynchron-Teilprogramm ausgeliefert.

ADJTCLT ist auf Unix-, Linux- und Windows-Systemen Bestandteil der Beispiel-Anwendung bzw. des Quick Start Kit.

Das Programm ermöglicht:

- Automatische Anpassung der Anzahl Prozesse der TAC-Klassen gemäß Tabelle. Diese Funktion wird immer ausgeführt.
- Einlesen einer neuen Tabelle mit Standardnamen, siehe Beispiel-Tabelle. Diese Funktion wird ausgeführt, wenn noch keine Tabelle eingelesen wurde oder wenn der Operationscode RF oder READFILE angegeben wird.
- Einlesen einer neuen Tabelle mit beliebigem Namen. Diese Funktion wird ausgeführt, wenn der Operationscode RF oder READFILE und ein Dateiname angegeben wird.
- Anpassen der aktuell erlaubten Anzahl Asynchron-Tasks. Diese Funktion wird ausgeführt, wenn der Operationscode MA=*ttt* oder MAXASYN=*ttt* angegeben wird, wobei *ttt* die Anzahl der gewünschten maximalen Anzahl ASYNTASKS ist.

  Da die Änderung der Anzahl der erlaubten Tasks für Asynchron-Verarbeitung keine Meldungen erzeugt, darf die Änderung nicht direkt über das Kommando KDCAPPL erfolgen, sondern muss über die durch dieses Programm angebotene Schnittstelle durchgeführt werden, damit die TAC-Klassen-Einstellungen angepasst werden.

> **i** Wird für das Teilprogramm nur ein Dialog-TAC generiert, so müssen alle Funktionen manuell angestoßen werden, d.h. wenn sich die Anzahl Tasks oder Asyntasks oder die Tabelle geändert hat, muss das Programm manuell aufgerufen werden.
>
> Wie das Programm automatisch über den Asynchron-TAC aufgerufen werden kann, ist im Abschnitt „ADJTCLT als MSGTAC- oder MSG-DEST-Teilprogramm" beschrieben.

TAC-Klassen-Tabelle erstellen

In der Tabelle legen Sie die Anzahl Tasks pro TAC-Klasse fest, und zwar in Abhängigkeit von:

- der Anzahl der laufenden Prozesse
- und der momentan eingestellten Anzahl Prozesse, die maximal für die Asynchron-Verarbeitung verwendet werden dürfen.

Die Tabelle muss als Textdatei gespeichert werden. Sie kann z.B. mit Microsoft Excel erstellt und dann als Tabstopp-getrennte Textdatei abgespeichert werden. Als Trennzeichen sind auch Leerzeichen erlaubt.

Beispiel-Tabelle

Auf allen Plattformen wird eine Beispiel-Tabelle mit dem folgenden Standardnamen ausgeliefert:

- BS2000-Systeme: ADJTABLE.TXT

Sie finden diese Beispiel-Tabelle in der Bibliothek SYSLIB.UTM.070.EXAMPLE. Damit das Teilprogramm diese Tabelle verwenden kann, müssen Sie sie auf die Benutzer-Kennung kopieren, unter der die UTM-Anwendung läuft. In UTM-Cluster-Anwendungen können alle Knoten- Anwendungen dieselbe Tabelle verwenden, wenn die Datei auf dem Shared Pubset liegt.

- Unix-, Linux- und Windows-Systeme: AdjTable.txt

  Auf Unix-, Linux- und Windows-Systemen ist die Tabelle Bestandteil der Beispiel-Anwendung bzw. des Quick Start Kit. Die Tabelle ist nach Installation von Beispiel-Anwendung bzw. Quick Start Kit in folgendem Unterverzeichnis von Beispiel-Anwendung bzw. Quick Start Kit zu finden:

  `utmsample/utm-c` (Unix- und Linux-Systeme)
  `utmsample\utm-c` (Windows-Systeme)

Aufbau der Tabelle

Für den Aufbau der Tabelle gelten folgende Regeln:

- Die Werte müssen als abdruckbare Zahlen angegeben werden.
- Die Tabelle kann als erste Zeile eine Überschriftenzeile enthalten.
- Jede weitere Zeile muss folgenden identischen Aufbau haben:
  - Spalte 1: Anzahl der laufenden Prozesse. Die maximale Anzahl Prozesse ist 240.
  - Spalte 2: Anzahl der momentan eingestellten Anzahl Prozesse, die maximal für die Asynchron-Verarbeitung verwendet werden dürfen.
  - ab Spalte 3: Prozess-Anzahl für jede TAC-Klasse in aufsteigender Reihenfolge, z.B.

    ```
    17 2 4 3 3 ... 0 0 0 1 1 1
    ```

    Spalte 3 entspricht TAC-Klasse 1, Spalte 4 TAC-Klasse 2 usw.

  Für jede dieser Zeilen gilt:

  - Die Anzahl der Prozesse in Spalte 1 muss größer sein als die Summe über die Anzahl der Prozesse aller Dialog-TAC-Klassen (1 - 8) plus die Anzahl der Asynchron-Prozesse. Je größer die Differenz ist, desto mehr Prozesse sind für die Erledigung von anderen Aufträgen frei.
  - Für Dialog-TAC-Klassen, die nicht verwendet werden, darf als Prozess-Anzahl auch 0 angegeben werden, obwohl der Minimalwert der Prozesse für Dialog-TAC-Klassen 1 ist. Der Grund: Ansonsten wäre die Minimalzahl Prozesse für die Anwendung 9 (8 Dialog-TAC-Klassen+ 1).

    Bitte beachten Sie, dass das Programm nicht überprüfen kann, ob diese Dialog-TAC-Klassen tatsächlich nicht verwendet werden.
  - Die Werte für die TAC-Klassen können auch weggelassen werden. Dann wird für Dialog-TAC-Klassen der Standardwert 0 und für Asynchron-TAC-Klassen der Standardwert '-' verwendet. '-' für Asynchron-TAC-Klassen bedeutet, dass die Task-Anzahl für diese TAC-Klasse nicht geändert wird.
- Die Prozess-Anzahl (Spalte1) muss innerhalb der Tabelle aufsteigend sortiert sein, während die Anzahl der Asynchron-Prozesse bei gleicher Prozess-Anzahl (Spalte 2) absteigend sortiert sein muss.
- Auf die Anzahl benötigter Prozesse hat nur die Anzahl maximal erlaubter Asynchron-Prozesse Einfluss, aber nicht die Anzahl der erlaubten Prozesse der einzelnen Asynchron-TAC-Klassen. Es wird nicht berücksichtigt, wenn die Anzahl der erlaubten Tasks der einzelnen Asynchron-TAC-Klassen kleiner ist als die Anzahl maximal erlaubter Asynchron-Tasks.

Beispiel

Ausschnitt aus einer Tabelle, mittels der bei bis zu 10 Prozessen immer ein Prozess, und ab 12 Prozessen immer zwei Prozesse freigehalten werden sollen, und in der nur die Dialog-TAC-Klassen 1, 2 und 3 verwendet werden.

Alle	Asyn	Tcl01	Tcl02	Tcl03	...	Tcl11	Tcl12	Tcl13	Tcl14	Tcl15	Tcl16
4	0	1	1	1	...	0	0	0	0	2	2
5	1	1	1	1	...	0	0	0	0	1	2
5	0	2	1	1	...	0	0	0	0	1	1
6	2	1	1	1							
6	1	2	1	1	...	0	0	0	0	0	1
6	0	2	2	1	...	0	0	0	0	0	0
...											
10	5	2	1	1	...	0	0	0	0	3	3
10	3	3	2	1	...	0	0	0	0	1	2
10	1	3	3	2	...	-	-	-	-	-	-
10	0	4	3	2							
12	7	1	1	1	...	0	0	0	0	3	4
12	3	3	2	2	...	0	0	0	0	1	2
12	1	4	3	2							

Das Programm beginnt die Suche mit dem letzten Tabelleneintrag und wählt den Tabelleneintrag aus, für den die beiden nachfolgenden Bedingungen gelten:

- Die Anzahl der laufenden Prozesse muss größer oder gleich der Zahl in Spalte 1 sein.
- Die Differenz aus der Anzahl der laufenden Prozesse und der aktuell eingestellten Maximalzahl Asynchron-Tasks muss größer oder gleich sein als Spalte 1 - Spalte 2.

  Diese Bedingung stellt sicher, dass die Anzahl der Prozesse, die mindestens für Dialogverarbeitung frei sind, tatsächlich größer ist als die Summe über die Anzahl der Prozesse in den Dialog-TAC-Klassen.

Wird ein Eintrag gefunden, dann berechnet das Programm Folgendes:

```
Anzahl verfügbarer Dialog-Prozesse = Spalte 1(Alle) - Spalte 2 (Asyn)
```

Beispiel:

- Bei 12 Prozessen und 7, 6, 5 oder 4 Asynchron-Prozessen wird folgende Zeile gewählt:

  ```
  12 7 1 1 1 ... 0 0 0 0 3 4
  ```

- Bei 12 Prozessen und 8 Asynchron-Prozessen wird folgende Zeile gewählt:

  ```
  6 2 1 1 1
  ```

  Von den 12 laufenden Prozessen sind immer 4 frei für Dialog-Verarbeitung, da maximal 8 Prozesse durch Asynchron-Verarbeitung belegt sind. Von diesen 4 Prozessen sind maximal 3 durch Dialog-Verarbeitung belegt (Summe über die Prozesse der Dialog-TAC-Klassen 1+1+1). Somit ist immer ein Prozess frei.

- Bei 12 Prozessen und 9 Asynchron-Prozessen wird kein passender Eintrag gefunden.

Wird ein passender Tabelleneintrag gefunden, so wird die Prozess-Anzahl der einzelnen TAC-Klassen gemäß der Tabelle angepasst.

In TAC-Klassen, die mit PGWT=YES generiert sind, darf die Anzahl Prozesse die generierte maximale Anzahl TASKS-IN-PGWT nicht überschreiten.

Überschreitet in TAC-Klassen, die mit PGWT=YES generiert sind, die Anzahl der Prozesse die maximal erlaubte Anzahl TASKS-IN-PGWT, so wird der kleinere Wert verwendet.

ADJTCLT als MSGTAC- oder MSG-DEST-Teilprogramm

Die Asynchron-Programm-Variante von ADJTCLT ist auch als MSGTAC- oder MSG-DEST-Programm nutzbar. Ein bestehendes MSGTAC-Programm kann (ggf. nach Anpassungen) in das Teilprogramm integriert werden. ADJTCLT ist jedoch nicht als Unterprogramm eines bestehenden MSGTAC-Programms verwendbar.

ADJTCLT als MSGTAC-Programm

Wird das ADJTCLT nur als MSGTAC-Programm generiert, dann stehen nur die Funktionen zur Verfügung, die keinen Operationscode benötigen, da das MSGTAC-Programm ausschließlich ereignisgesteuert abläuft und nicht über einen TAC aufgerufen werden kann.

In diesem Fall muss ein eigener Anwendungs-Meldungsmodul existieren und die Meldungen K052, K056 und K058 müssen das Meldungsziel MSGTAC haben.

ADJTCLT als MSGDEST-Programm

Für meldungsgesteuerte Verarbeitung kann der Asynchron-TAC auch als MSG-DEST generiert (MSG-DEST USER-DEST-1/2/3/4, NAME=, DEST-TYPE=TAC, ...) und im Anwender- Meldungsmodul als USER-DEST für die Meldung K052, K056, K058 verwendet werden.

Allerdings läuft ein meldungsgesteuertes Teilprogramm nur dann unabhängig von der Anzahl Asynchron-Prozesse und der TAC-Klassen-Steuerung, wenn es als MSGTAC-Programm generiert ist. Ansonsten gilt:

- Für Asynchron-Verarbeitung muss immer mindestens ein Prozess zugelassen sein.
- In der TAC-Klasse des Teilprogramms muss mindestens ein Prozess zugelassen sein.

Hinweise zur Generierung:

Das Teilprogramm muss im KDCDEF-Lauf wie folgt definiert werden:

```
PROGRAM ADJTCLT, COMP=ILCS (BS2000-Systeme)
PROGRAM ADJTCLT, COMP=C (Unix-, Linux- und Windows-Systeme)
```

Generierung als Dialog- und Asynchron-TAC:

```
TAC ADJTCLT, PROGRAM=ADJTCLT, ADMIN=YES, TYPE=D
TAC ADJTCLTA, PROGRAM=ADJTCLT, ADMIN=YES, TYPE=A
```

Generierung als MSGTAC:

```
TAC KDCMSGTC, PROGRAM=ADJTCLT, ADMIN=YES, TYPE=A
```

Der Asynchron-TAC ADJTCLTA kann wie folgt als MSG-DEST für z.B. USER-DEST-1 generiert werden:

```
MSG-DEST USER-DEST-1, NAME=ADJTCLTA, DEST-TYPE=TAC , MSG-FORMAT=PRINT
```

Auf Unix-, Linux- und Windows-Systemen können Sie die KDCDEF-Anweisungen auch aus der Beispiel-Anwendung bzw. dem Quick Start Kit in die Generierung der UTM-Anwendung übernehmen.

Hinweise zum Binden:

Auf BS2000-Systemen kann das Teilprogramm ADJTCLT über eine RESOLVE-BY-AUTO-Anweisung in das Anwendungsprogramm eingebunden werden. Dadurch wird auch die Routine ERRCHCK implizit mit eingebunden.

Auf Unix-, Linux- und Windows-Systemen wird das Teilprogramm ADJTCLT automatisch in die Beispiel-Anwendung bzw. in das Quick Start Kit eingebunden.

14.3 CALLUTM - Tool für Administration und Client-Server-Kommunikation (BS2000-Systeme)

CALLUTM ist ein UPIC-Client auf BS2000, der mit UTM-Anwendungen kommuniziert, die entweder auf dem gleichen BS2000-Rechner oder einem anderen Rechner ablaufen. CALLUTM kann mit UTM-Anwendungen kommunizieren unabhängig davon, unter welchem Betriebssystem die UTM-Anwendungen ablaufen.

Mit CALLUTM können Sie aus einer BS2000-Task heraus Services in einer UTM-Anwendung starten, Daten an die Services übergeben und Nachrichten von den Services empfangen. Die Ausgabe der Nachrichten erfolgt im Line-Mode. CALLUTM ist sowohl im Dialog- als auch im Batch-Betrieb ablauffähig, d.h. es kann auch in prozeduralen Umgebungen innerhalb einer BS2000-Task eingesetzt werden.

Aus diesem Grund können Sie CALLUTM insbesondere für die zentrale Administration von lokalen und fernen UTM-Anwendungen nutzen. Sie können mit CALLUTM UTM-Administrationskommandos absetzen und Administrationsprogramme in den UTM-Anwendungen starten.

Dazu müssen Sie die Generierungen der administrierten UTM-Anwendungen anpassen, siehe Abschnitt „Generierung".

Für die folgende Beschreibung von CALLUTM werden Kenntnisse über UPIC auf BS2000-Systemen vorausgesetzt, siehe Handbuch „openUTM-Client für das Trägersystem UPIC“.

14.3.1 Generierung

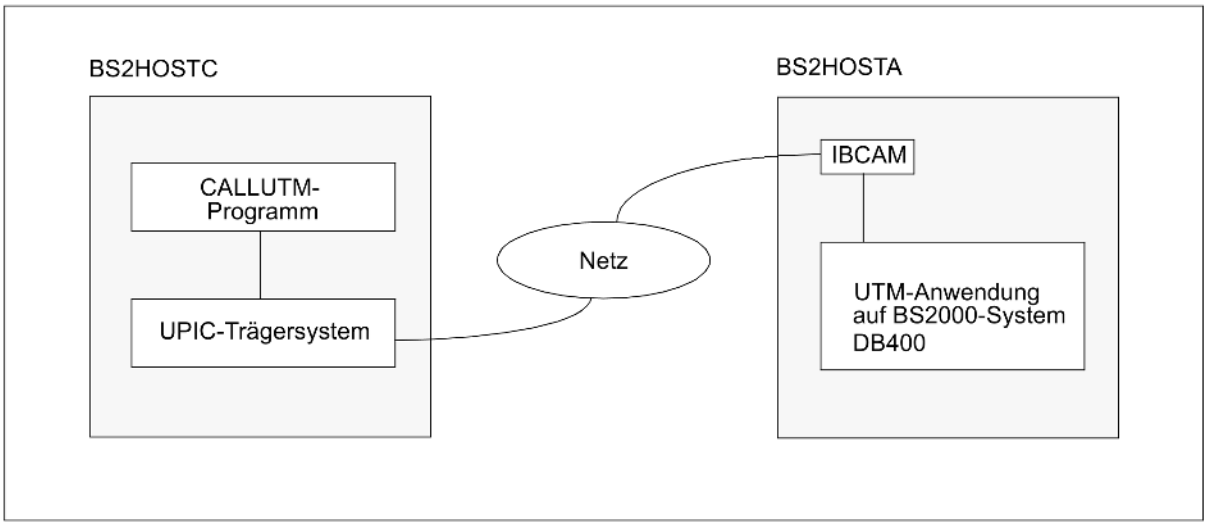
Um CALLUTM für die Administration von UTM-Anwendungen auf BS2000-Systemen einzusetzen, müssen Sie Folgendes tun:

- Im lokalen BS2000-System legen Sie in der „side information file“ für das Trägersystem UPIC, der sog. UPICFILE, Einträge für die UTM-Anwendungen an (siehe Handbuch „openUTM-Client für das Trägersystem UPIC“).
- In jeder zu administrierenden UTM-Anwendung müssen Sie PTERM-Einträge und LTERM-Partner in die jeweilige Konfiguration eintragen bzw. einen LTERM-Pool generieren, über den sich CALLUTM anschließen kann.
- In jeder zu administrierenden UTM-Anwendung müssen Sie mindestens eine Benutzerkennung mit Administrationsberechtigung erzeugen (siehe Beispiel unten). Diese Benutzerkennung (zusammen mit dem zugehörigen Passwort) muss CALLUTM dann beim Einrichten der Conversation an die UTM-Anwendung übergeben. Dazu stehen in der CALLUTM-Anweisung CREATE-CONFIGURATION die Operanden USER-ID und PASSWORD zur Verfügung (siehe Abschnitt "CREATE-CONFIGURATION" im Abschnitt "Beschreibung der CALLUTM-Programm-Anweisungen (BS2000- Systeme)".

> **i** Sie können auch dem LTERM-Partner, über den sich CALLUTM an die UTM-Anwendung anschließt, direkt eine Benutzerkennung mit Administrationsberechtigung zuordnen (LTERM ...,USER=). CALLUTM muss dann keine Benutzerkennung angeben und ist nach dem Aufbau der Verbindung zur UTM-Anwendung administrationsberechtigt. Beachten Sie, dass diese Vorgehensweise jedoch den Zugangsschutz der UTM-Anwendung herabsetzt.

Beispiel

Das Programm CALLUTM auf dem BS2000-Rechner BS2HOSTC soll mit der Anwendung DB400 auf dem BS2000-Rechner BS2HOSTA kommunizieren.



Beispielkonfiguration für den Einsatz des Programms CALLUTM

KDCDEF-Generierung der UTM-Anwendung DB400 am Rechner BS2HOSTA

CALLUTM soll sich auf zwei Arten an die UTM-Anwendung anschließen können:

- über einen LTERM-Pool. Dazu muss in der UTM-Anwendung eine TPOOL-Anweisung abgesetzt werden.
- über einen LTERM-Partner, der explizit für die Zusammenarbeit mit CALLUTM generiert wird. Dazu müssen in der UTM-Anwendung für CALLUTM eine PTERM-Anweisung und eine LTERM-Anweisung abgesetzt werden.

Es wird eine Benutzerkennung (ADMUPCT0) mit Administrationsberechtigung generiert, mit der sich CALLUTM beim Aufbau der Conversation bei der UTM-Anwendung anmelden kann.

```
*****
*- BCAMAPPL FUER DEN ANSCHLUSS VON CALLUTM UEBER LTERM-POOL
*****
BCAMAPPL DB4UPAP0,T-PROT=ISO
*****
*- BCAMAPPL FUER DEN ANSCHLUSS VON CALLUTM UEBER EINEN EIGENEN LTERM-PARTNER
*****
BCAMAPPL DB4UPAT0,T-PROT=ISO
*****
*- LTERM-POOL FUER DEN ANSCHLUSS VON CALLUTM ----------------------*
*****
TPOOL BCAMAPPL=DB4UPAP0,KSET=ALLKEYS,LTERM=UPCP0#0,NUMBER=9,      -
PRONAM=BS2HOSTC,PROTOCOL=N,PTYPE=UPIC-R
*****
*- PTERM-ANWEISUNG UND LTERM-PARTNER FUER CALLUTM DEFINIEREN -----*
*****
PTERM UPCPT#T0, PRONAM=BS2HOSTC, PTYPE=UPIC-R,                   -
                BCAMAPPL=DB4UPAT0, PROTOCOL=N, LTERM=UPCLT#T0
LTERM UPCLT#T0, KSET=ALLKEYS, RESTART=N
*****
*- ADMINISTRATIONSBERECHTIGTE BENUTZERKENNUNG --------------------*
*****
USER ADMUPCT0, PERMIT=ADMIN, PASS=(ADMT0   ,D)
*****
*-----------------------------------------------------------------*
*****
```

Einträge in der UPICFILE

CALLUTM an die UTM-Anwendung anschließen

- CALLUTM schließt sich über den LTERM-Partner UPCLT#T0 an die UTM-Anwendung DB400 auf dem Rechner BS2HOSTA an, wenn Sie beim Aufbau der Verbindung in CALLUTM Folgendes angeben:

  ```
  local name = UPCPT#T0
  symbolic destination name = DBSADMT0
  ```

  „UPCPT#T0“ wird beim Verbindungsaufbau als Client-Name (PTERM-Name) an openUTM übergeben.

  Der symbolische Partnername DBSADMT0 wird mit dem Partnernamen DB4UPAT0 aus der UPICFILE verknüpft. Das ist der Name der UTM-Anwendung DB400, der bei der KDCDEF-Generierung in BCAMAPPL angegeben wurde.

- CALLUTM schließt sich über den LTERM-Pool an die UTM-Anwendung an, wenn Sie beim Aufbau der Verbindung Folgendes angeben:

  ```
  local name = locname
  symbolic destination name = DBS0POOL oder DBS1POOL
  ```

  Für *locname* ist ein alphanumerischer Name (bis zu 8 Byte lang) anzugeben, mit dem sich CALLUTM beim Transportsystem anmeldet. Er wird beim Verbindungsaufbau als Client-Name an openUTM übergeben und für die Dauer der Verbindung einem LTERM-Partner des LTERM-Pools zugeordnet.

  Der symbolische Partnername DBS0POOL bzw. DBS1POOL wird mit dem Partnernamen DB4UPAP0 aus der UPICFILE verknüpft. Das ist der Name der UTM-Anwendung DB400, der bei der KDCDEF-Generierung in BCAMAPPL angegeben wurde.

In einer BS2000-Task CALLUTM als Administratorprogramm aufrufen

Sie können CALLUTM über das SDF-Kommando START-CALLUTM aufrufen. Dieses Kommando ist im SDF-Anwendungsbereich UTM abgelegt. Weitere Informationen finden Sie im openUTM-Handbuch „Einsatz von UTM-Anwendungen auf BS2000-Systemen“, Abschnitt „UTM-Tools aufrufen“.

Die Kommunikation zum Server kann sowohl verschlüsselt als auch unverschlüsselt erfolgen.

Beispiel (Aufruf über START-CALLUTM)

Im folgenden Beispiel soll CALLUTM die UTM-Anwendung DB400 auf dem Rechner BS2HOSTA beenden. Dazu müssen Sie CALLUTM starten und sich mit der administrationsberechtigten Benutzerkennung ADMUPCT0 bei der UTM-Anwendung DB400 anmelden. Zum Beenden der Anwendung setzen Sie das UTM-Administrationskommando KDCSHUT NORMAL ab.

```
/START-CALLUTM
//CREATE-CONFIGURATION LOCAL-NAME=UPCPT#T0,                              -
/                              SYMB-DEST-NAME=DBSADMT0
//SELECT-SERVICE SERVICE-NAME=KDCSHUT, SERVICE-DATA='NORMAL'
//END
```

Die Kommunikation mit dem Server wird standardmäßig über eine Socket-Verbindung abgewickelt. Soll CMX verwendet werden, verwenden Sie folgendes Kommando.

```
/START-CALLUTM TRANSPORT-SYSTEM=*CMX
```

14.3.2 Beschreibung der CALLUTM-Programm-Anweisungen

Die Programm-Anweisungen von CALLUTM werden über die SDF-Benutzeroberfläche des BS2000-Systems gelesen und vom Kommandoprozessor SDF verarbeitet. Das Programm CALLUTM hat neben den SDF-Standardanweisungen die in der folgenden Tabelle aufgelisteten Programm-Anweisungen.

Nr.	Programm-Anweisung	Funktion
1.	CREATE-CONFIGURATION	Umgebung für den Programmablauf definieren und die Verbindung zur UTM-Anwendung auswählen. Diese Anweisung muss als erste Anweisung abgesetzt werden.
2.	SELECT-SERVICE	Einen Service (Transaktionscode) in der UTM-Anwendung starten. Es kann eine Nachricht (Operanden oder Parameterwerte) mitgegeben werden.
3.	CONTINUE-SERVICE	Einen noch nicht beendeten Service fortsetzen. Die Anweisung dient dazu, bei Services mit mehreren Verarbeitungsschritten, nach Beendigung eines Verarbeitungsschritts den folgenden Verarbeitungsschritt zu starten. Es kann eine Nachricht mitgegeben werden.
4.	DEALLOCATE-CONVERSATION	Die Conversation zur UTM-Anwendung beenden. Ein in der UTM-Anwendung noch offener Service, der zu dieser Conversation gehört, wird abnormal beendet.
5.	SHOW-CONFIGURATION	Programmablauf-Umgebung, die mit CREATE-CONFIGURATION oder MODIFY-CONFIGURATION eingestellt wurde, anzeigen lassen.
6.	MODIFY-CONFIGURATION	Programmablauf-Umgebung, die mit CREATE-CONFIGURATION eingestellt wurde, modifizieren.
7.	CALLUTM-ERROR-STEP	Verarbeitung der Anweisungen für CALLUTM innerhalb einer Prozedur oder im Batch steuern.

CREATE-CONFIGURATION muss immer die erste Anweisung nach dem Programmstart sein. Insbesondere muss CREATE-CONFIGURATION vor SELECT-SERVICE abgesetzt werden.
Die Anweisungen 5. und 6. können an einer beliebigen Stelle zwischen CREATE-CONFIGURATION und dem Ende des Programmlaufs abgesetzt werden.

Die Anweisungen sind im Folgenden in alphabetischer Reihenfolge detailliert beschrieben.

Metasyntax

Bei der Beschreibung der Anweisungen wird die SDF-Syntax-Darstellung verwendet. In der folgenden Tabelle sind die Darstellungselemente beschrieben, die zusätzlich zu der im Abschnitt "Darstellungsmittel" beschriebenen Metasyntax verwendet werden.

Kennzeichnung	Bedeutung	Beispiele
< >	Spitze Klammern kennzeichnen Variablen, deren Wertevorrat durch Datentypen und ihre Zusätze beschrieben wird.	**POSITION** = <integer 1..256>
/	Der Schrägstrich trennt alternative Operandenwerte.	**SET-TEST-MODE** = ***NO** / ***YES**
(...)	Runde Klammern kennzeichnen Operandenwerte, die eine Struktur einleiten.	,**SET-SERVICE-JV** = ***YES** (...) / ***NO**
Einrückung	Die Einrückung kennzeichnet die Abhängigkeit zu dem jeweils übergeordneten Operanden.	,**SET-SERVICE-JV** = ***NO** / ***YES**(...) ***YES**(...) | **JV-IDENTIFICATION** = **...** |
| | |	Der Strich kennzeichnet zusammengehörende Operanden einer Struktur. Sein Verlauf zeigt Anfang und Ende einer Struktur an. Innerhalb einer Struktur können weitere Strukturen auftreten. Die Anzahl senkrechter Striche vor einem Operanden entspricht der Strukturtiefe.	, **SET-SERVICE-JV** = ***NO** / ***YES** (...) ***YES** (...) | **JV-ID** = ***JV-NAME** (...) /... | ***JV-NAME** (...) | | **JV-NAME** =...
Kurzname:	Der darauf folgende Name ist ein garantierter Aliasname des Anweisungsnamens.	Kurzname: **CONFATTR**

CALLUTM-ERROR-STEP

Die Anweisung CALLUTM-ERROR-STEP steuert die Verarbeitung der Anweisungen für CALLUTM innerhalb einer Prozedur oder im Batch:

Wenn während des Programmlaufs ein Fehler auftritt (kein SDF-Syntax-Fehler), z.B. die adressierte openUTM-Anwendung läuft nicht mehr, liest CALLUTM die nächsten Anweisungen so lange von SYSDTA bis die Anweisung CALLUTM-ERROR-STEP gefunden wird. Dann wird der Programmlauf mit der nächsten Anweisung fortgesetzt. Andernfalls wird CALLUTM beendet.

Die Anweisung hat keine Operanden.

Beispiel

```
//CREATE-CONFIGURATION ...
```

```
// ...
// ...
//SELECT-SERVICE SERVICE-NAME = KDCINF, SERVICE-DATA = C'STAT'       (1)
//SELECT-SERVICE SERVICE-NAME = KDXINF, SERVICE-DATA = C'USER'       (2)
//SELECT-SERVICE SERVICE-NAME = KDCINF, -                            (3)
// SERVICE-DATA = C'USER,L=KDCCON'
//CALLUTM-ERROR-STEP                                                (4)
//SELECT-SERVICE SERVICE-NAME = KDCINF, SERVICE-DATA = C'TAC'        (5)
```

Erläuterung:

Anweisung (1) wird ausgeführt.
Anweisung (2) bringt Fehler, da TAC KDXINF nicht definiert ist.
Anweisung (3) wird nicht ausgeführt.
Die Verarbeitung wird mit Anweisung (5) wieder aufgenommen.

CONTINUE-SERVICE

Mit CONTINUE-SERVICE können Sie einen Vorgang in der UTM-Anwendung, der mit SELECT-SERVICE angestoßen wurde und aus mehreren Schritten besteht, fortsetzen. CONTINUE-SERVICE geben Sie an, wenn der Service nach Beendigung eines Dialog-Schritts eine Nachricht an CALLUTM gesendet hat und der Vorgang noch nicht beendet ist, weil noch weitere Verarbeitungsschritte durchgeführt werden müssen/können. Für den folgenden Verarbeitungsschritt können Daten an den Service übergeben werden.

CONTINUE-SERVICE

SERVICE-DATA = **<u>*NO</u>** / list-poss(42):<c-string -with-lower-case 1..1800>

,**SET-SERVICE-JV** = **<u>*NO</u>** / ***YES**(...)

***YES**(...)

| **JV-IDENTIFICATION** = ***JV-NAME**(...) / ***LINK-NAME**(...)

| ***JV-NAME**(...)

| | **JV-NAME**=<full-filename-without-generation-version 1..54>

| | ,**POSITION** = **<u>1</u>** / <integer 1..256>

| | ,**LENGTH** = **<u>*REST</u>** / <integer 1..256>

| ***LINK-NAME**

| | **LINK-NAME** =< alphanum-name 1..7>

| | ,**POSITION** = **<u>1</u>** / <integer 1..256>

| | ,**LENGTH** = **<u>*REST</u>** / <integer 1..256>

| ,**PASSWORD** = **<u>*NONE</u>** /< c-string 1..4> / <x-string 1..8>

| ,**VALUE** = **<u>*RECEIVE-MSG</u>** (...) / <c-string-with-lower-case 1..256> / <x-string 1..512>

| ***RECEIVE-MSG**(...)

| | **POSITION** = **<u>1</u>** / <integer 1..4000>

Operandenbeschreibung siehe Anweisung SELECT-SERVICE.

Beispiel

In diesem Beispiel wird auf das Beispiel-Administrationsprogramm SUSRMAX Bezug genommen, das zusammen mit openUTM ausgeliefert wird (siehe Abschnitt „Beispielprogramme"). Der Dialog mit SUSRMAX läuft in den folgenden Teilschritten ab:

1. SUSRMAX wird gestartet und liefert eine Nachricht zurück, die zur Auswahl einer Funktion auffordert:

```
-> //SELECT-SERVICE SERVICE-NAME=SUSRMAX
   <date: 10-19-2017 time: 10:18:55
   application: DB400    host: BS2HOSTA tac: SUSRMAX
   ---------------------------------------------------------------
```

```
   available commands:
   0 = end                          |  1 = show-connected-users
   2 = show-users-in-conversation   |  3 = show-changeable-max-values
   4 = change-max-values            |
   please make a selection
```

2. Der Vorgang wird mit CONTINUE-SERVICE fortgesetzt, es wird die Funktion „1 = show-connected-users“ ausgewählt (SERVICE-DATA='1'). openUTM liefert die geforderten Informationen zurück.

```
-> //CONTINUE-SERVICE SERVICE-DATA='1'
<-   ...
     ... Ausgabe
     ...
```

3. Die UTM-Nachricht zur Auswahl einer weiteren Funktion soll erneut ausgegeben werden.

```
-> //CONTINUE-SERVICE
<-   ...
     ... Ausgabe der Funktionsauswahl-Nachricht wie unter 1.
     ...
```

4. Der Vorgang wird mit CONTINUE-SERVICE fortgesetzt, es wird die Funktion „2 = show-users-in-conversation“ ausgewählt (SERVICE-DATA='2'). openUTM liefert die geforderten Informationen zurück.

```
-> //CONTINUE-SERVICE SERVICE-DATA='2'
<-   ...
     ... Ausgabe
     ...
```

5. SUSRMAX soll beendet werden (Funktion „0 = end“).

```
-> //CONTINUE-SERVICE SERVICE-DATA='0'
   <date: 10-19-2017 time: 10:20:18
   application: DB400    host: BS2HOSTA tac: SUSRMAX
   ---------------------------------------------------------------
   conversation terminated
   ---------------------------------------------------------------
-> //
```

CREATE-CONFIGURATION

Mit der Anweisung CREATE-CONFIGURATION wird die Umgebung für den Programmablauf definiert und die Verbindung zur UTM-Anwendung ausgewählt, d.h. Sie legen Folgendes fest:

- wie sich das Programm beim Trägersystem UPIC anmeldet
- zu welcher UTM-Anwendung eine Verbindung aufgebaut werden soll
- welche UTM-Benutzerkennung beim Aufbau der Conversation übergeben wird
- ob und in welchem Umfang eine Protokolldatei geführt wird
- ob der UPICTRACE mitlaufen soll.

CREATE-CONFIGURATION muss die erste Anweisung nach dem Programmstart sein. Wird CREATE-CONFIGURATION innerhalb des Programmlaufs wiederholt abgesetzt, dann werden eventuell offene Protokolldateien geschlossen und offene Services zurückgesetzt. Es wird intern ein DEALLOCATE durchgeführt.

Die mit CREATE-CONFIGURATION eingestellten Werte können Sie innerhalb eines Programmlaufs mit MODIFY-CONFIGURATION ändern.

CREATE-CONFIGURATION Kurzname: **CONFATTR**

LOCAL-NAME = <alphanum-name 1..8>

,**SYMB-DEST-NAME** = <alphanum-name 8..8>

USER-ID = **<u>*NONE</u>** / <alphanum-name 1..8>(...) / <c-string_1..8_with-low> / < x-string 1..16>

<alphanum-name 1..8>(...) / <c-string_1..8_with-low> / < x-string 1..16>

| **PASSWORD** = **<u>*NONE</u>** /< c-string 1..16 >/< x-string 1..32>

,**WRITE-LOGGING-FILE** = **<u>*NO</u>** / ***YES** (...)

***YES**(...)

| **LOGGING-FILENAME** = <full-filename-without-generation-version 1..54>

| ,**OPEN-MODE** = ***REPLACE** / **<u>*EXTEND</u>**

| ,**LOGGING-INFO** = **<u>*ALL</u>** / ***SEND** / ***RECEIVE**

| ,**RECORD-LENGTH** = **<u>*STD</u>** / <integer 1..252>

,**WRITE-UPIC-TRACE** = **<u>*NO</u>** / ***YES**

,**CONFIGURATION-ID** = **<u>1</u>** / <integer 1..1>

,**SET-TEST-MODE** = **<u>*NO</u>** / ***YES**

,SET-ENCRYPTION-LEVEL = **<u>*NO</u>** / <integer 1..4>

LOCAL-NAME = <alphanum-name 1..8>

Lokaler Name, mit dem sich CALLUTM beim Trägersystem UPIC anmeldet. Der Name muss in der UPICFILE definiert sein.

SYMB-DEST-NAME = <alphanum-name 8..8>

Symbolischer Partnername der UTM-Anwendung, zu der eine Verbindung aufgebaut werden soll. Der Name muss in der UPICFILE definiert sein.

USER-ID =

UTM-Benutzerkennung, die zum Einrichten der Conversation benutzt wird.

*NONE

Es wird ohne Security-Funktionen gearbeitet.

<alphanum-name 1..8>( ) / <c-string_1..8_with-low> / < x-string 1..16>

UTM-Benutzerkennung. Die Benutzerkennung muss in der UTM-Anwendung existieren.

PASSWORD = *NONE
Der Benutzerkennung in USER-ID ist kein Passwort zugeordnet.

PASSWORD = <c-string 1..16> oder <x-string 1..32>
Ist der Benutzerkennung in USER-ID ein Passwort zugeordnet, dann muss dieses Passwort hier als Character-String (c-string) oder hexadezimale Zeichenfolge (x-string) angegeben werden.

WRITE-LOGGING-FILE =

Bestimmt, ob und in welchem Umfang der Datenfluss vom Client zum Server und zurück (Daten der Services) protokolliert werden soll.

*NO

Es wird nicht protokolliert.

*YES ( )

Es wird protokolliert.

LOGGING-FILENAME = <full-filename-without-gen-vers 1..54>
Name der Protokolldatei.

OPEN-MODE = *EXTEND
Protokolldatei wird fortgeschrieben, falls vorhanden. Falls die Protokolldatei noch nicht existiert, wird sie neu angelegt.

OPEN-MODE = *REPLACE
Protokolldatei wird überschrieben, falls vorhanden, bzw. neu angelegt.

LOGGING-INFO = *SEND
Nur die an einen Service der UTM-Anwendung gesendeten Daten werden protokolliert.

LOGGING-INFO = *RECEIVE
Nur die Daten, die CALLUTM von einem Service der UTM-Anwendung empfängt, werden protokolliert.

LOGGING-INFO = *ALL
Es werden alle Daten protokolliert, die CALLUTM mit einem Service der UTM-Anwendung austauscht, d. h. sowohl die Sende- als auch Empfangsdaten.

RECORD-LENGTH = *STD / <integer 1..252>
Satzlänge, mit der in die Protokolldatei geschrieben wird. Ein neuer Satz beginnt, wenn im Sende- oder Empfangsbereich „newline“ gefunden wird. „newline“ selbst wird nicht protokolliert.
*STD steht für eine Satzlänge von 79 Byte.

WRITE-UPIC-TRACE =

gibt an, ob der UPIC-Trace gestartet wird.

*NO

Der UPIC-Trace wird nicht eingeschaltet.

*YES

Der UPIC-Trace wird eingeschaltet.
Existiert kein Kettungsname *UPICTRA zu einer Jobvariablen, so wird die Jobvariable JV.UPICTRACE. CALLUTM mit dem Wert „-SX“ neu eingerichtet. Ihr wird der Kettungsname *UPICTRA zugewiesen.

CONFIGURATION-ID = 1

Dient der Identifikation der Konfiguration. Es ist nur der Wert 1 zulässig.

SET-TEST-MODE=

Testmodus ein- bzw. ausschalten.

*NO

Der Testmodus wird nicht eingeschaltet. Es werden keine UPIC-Aufrufe nach SYSOUT ausgegeben.

*YES

Testmodus wird eingeschaltet. Alle UPIC-Aufrufe von CALLUTM werden auf SYSOUT ausgegeben.

SET-ENCRYPTION-LEVEL=

Gibt an, ob und wie auf der Verbindung Daten verschlüsselt werden sollen.

Ist für die Client-Verbindung auf der Server-Seite Verschlüsselung generiert, muss hier der entsprechende Encryption-Level aus der Generierung angegeben werden.

Ist auf der Server-Seite für einen TAC Verschlüsselung generiert, der hier aufgerufen werden soll, muss auf der Client-Seite ebenfalls der Encryption-Level angegeben werden.

*NO

Keine Verschlüsselung.

<integer 1..4>

Encryption-Level aus der Generierung

Beispiel

```
//CREATE-CONFIGURATION LOCAL-NAME=UPCPT#T0,SYMB-DEST-NAME=DBSADMT0, -
//    WRITE-LOGGING-FILE=*YES(L-F=LOG.CALLUTM)
```

DEALLOCATE-CONVERSATION

Mit dieser Anweisung wird die Conversation zur Partner-Anwendung beendet. Ein eventuell noch offener Service in der UTM-Anwendung wird abnormal beendet.

DEALLOCATE-CONVERSATION
CONFIGURATION-ID = **1** / <integer 1..1>

CONFIGURATION-ID muss nicht angegeben werden (Standardeinstellung). Es darf nur CONFIGURATION-ID=1 angegeben werden.

Beispiel

```
//DEALLOCATE-CONVERSATION
```

MODIFY-CONFIGURATION

Mit MODIFY-CONFIGURATION können Werte der aktuell eingestellten Programmablauf-Umgebung modifiziert werden, die mit CREATE-CONFIGURATION oder durch ein vorheriges MODIFY-CONFIGURATION gesetzt wurden.

MODIFY-CONFIGURATION	Kurzname: **MODATTR**

LOCAL-NAME = *UNCHANGED / <alphanum-name 1..8>

,**SYMB-DEST-NAME** = ***UNCHANGED** / <alphanum-name 8..8>

,USER-ID = *UNCHANGED / <alphanum-name 1..8>

<alphanum-name 1..8>(...)

| **PASSWORD = *UNCHANGED / *NONE /< c-string 1..8> / <x-string 1..16>**

,**WRITE-LOGGING-FILE** = ***UNCHANGED** / ***YES** (...)

***YES(...)**

| **LOGGING-FILENAME = *UNCHANGED**

| ,**OPEN-MODE** = ***UNCHANGED** / ***REPLACE** / ***EXTEND**

| ,**LOGGING-INFO** = ***UNCHANGED** / ***ALL** / ***SEND** / ***RECEIVE**

| ,**RECORD-LENGTH** = ***UNCHANGED** / ***STD** /< integer 1..252>

,**WRITE-UPIC-TRACE** = ***UNCHANGED** / ***NO** / ***YES**

,**CONFIGURATION-ID** = **1** / <integer 1..1>

,**SET-TEST-MODE** = ***UNCHANGED** / ***NO** / ***YES**

,SET-ENCRYPTION-LEVEL = *UNCHANGED / *NO

Operandenbeschreibung siehe Anweisung CREATE-CONFIGURATION.

Beispiel

Es wird die Protokollierung ausgeschaltet.

```
-> //MOD-CONF WRITE-LOGGING-FILE=*NO
<- CUA0050: configuration modified
-> //
```

SELECT-SERVICE

Mit SELECT-SERVICE wird in der UTM-Anwendung ein Service gestartet. Es können Daten mitgegeben werden, d. h. Operanden und Parameterwerte, die der Service für die Bearbeitung benötigt. Zudem kann eine Jobvariable

bestimmt werden, die die Empfangs-Nachricht, einen Teil davon oder einen angegebenen String nach Ausführung der Anweisung aufnimmt. Ist die Jobvariable noch nicht katalogisiert, wird sie neu angelegt.

Wird mit SELECT-SERVICE ein Service aufgerufen, der in mehreren Verarbeitungsschritten abläuft und zwischen den einzelnen Verarbeitungsschritten Dialog-Nachrichten an CALLUTM übergibt, dann sind nach dem Aufrufen des Services und vor dem Beenden des Vorgangs außer den SDF-Standardanweisungen nur die folgenden Anweisungen erlaubt:

- CONTINUE-SERVICE
  zum Fortsetzen des Service nach dem Empfang einer Dialog-Nachricht
- DEALLOCATE-CONVERSATION
  zum Abbruch der Verbindung und zur (abnormalen) Beendigung des Service in der UTM-Anwendung.
- SHOW-CONFIGURATION
  zum Lesen der aktuell eingestellten Konfigurationsdaten.

SELECT-SERVICE

SERVICE-NAME = <alphanum-name 1..8>/<c-string-with-lower-case>

,**SERVICE-DATA** = **<u>*NO</u>** / list-poss(42): <c-string -with-lower-case 1..1800>

,**SET-SERVICE-JV** = **<u>*NO</u>** / ***YES**(...)

***YES**(...)

| **JV-ID**ENTIFICATION = ***JV-NAME**(...) / ***LINK-NAME**(...)

| ***JV-NAME**(...)

| | **JV-NAME**=<full-filename-without-generation-version 1..54>

| | ,**POSITION** = **<u>1</u>** / <integer 1..256>

| | ,**LENGTH** = **<u>*REST</u>** / <integer 1..256>

| ***LINK-NAME**(...)

| | **LINK-NAME** =< alphanum-name 1..7>

| | ,**POSITION** = **<u>1</u>** / <integer 1..256>

| | ,**LENGTH** = **<u>*REST</u>** / <integer 1..256>

| ,**PASSWORD** = **<u>*NONE</u>** /< c-string 1..4> / <x-string 1..8>

| ,**VALUE** = **<u>*RECEIVE-MSG</u>** (...) / <c-string-with-lower-case 1..256> / <x-string 1..512>

| ***RECEIVE-MSG**(...)

| | **POSITION** = **<u>1</u>** / <integer 1..4000>

SERVICE-NAME = <alphanum-name 1..8>/<c-string-with-lower-case>

der Transaktionscode, mit dem ein Vorgang in der UTM-Anwendung gestartet werden soll. Das kann auch ein Administrationskommando oder ein anderer Administrations-TAC sein.

SERVICE-DATA =

Nachricht, die an den Service in der UTM-Anwendung übergeben werden soll.

*NO

Es werden keine Daten mitgegeben.

list-poss(42): <c-string-with-lower-case 1..1800>

Nachricht, die an den fernen Service übergeben werden soll. Die Nachricht muss als C-String, d.h. in Hochkommas eingeschlossen, übergeben werden.
Sie können hier auch eine Liste von C-Strings übergeben. Die einzelnen Listenelemente werden als Teilnachrichten gesendet und auch als Teilnachrichten vom Server empfangen.
Anzahl C-Strings: maximal 42
Gesamtlänge der Liste: maximal 1800 Zeichen

SET-SERVICE-JV =

Versorgen einer Jobvariablen.

*NO

Es wird keine Jobvariable versorgt.

*YES ( )

Es wird eine Jobvariable versorgt.

JV-IDENTIFICATION = *JV-NAME ( )
Die Jobvariable wird über ihren Namen angesprochen

JV-NAME = <full-filename-without-genereration-version 1..54>
Name der Jobvariablen. Ist die Jobvariable noch nicht katalogisiert, dann wird sie neu angelegt.

POSITION = 1
Die Jobvariable wird ab Spalte 1 gesetzt.

POSITION = <integer 1..256>
Die Jobvariable wird ab der angegebenen Spalte gesetzt.

LENGTH = *REST
Die Jobvariable kann ab POSITION in voller Länge gesetzt werden

LENGTH = <integer 1..256>
Die Jobvariable wird in der angegebenen Länge gesetzt(abhängig von Eingabewert und Startposition).

JV-IDENTIFICATION = *LINK-NAME ( )
Die Jobvariable wird über einen Linknamen angesprochen, der vor Ausführung der Anweisung gesetzt worden sein muss (z.B. bei Beginn des Programmlaufs).

LINK-NAME = <alphanum-name 1..7>
Gesetzter Linkname der Jobvariablen.

POSITION = wie oben bei *JV-NAME ( )

LENGTH = wie oben bei *JV-NAME ( )

PASSWORD = *NONE
Der Zugriff auf die Jobvariable ist nicht über ein Passwort geschützt.

PASSWORD = <c-string 1..4> / <x-string 1..8>
Passwort für den Zugriff (lesend und schreibend) auf die Jobvariable.

VALUE = *RECEIVE-MSG (POSITION = 1/<integer 1..4000>)
Die Jobvariable wird mit den von der UTM-Anwendung empfangenen Daten besetzt (entsprechend der oben definierten Position und Länge). In POSITION geben Sie an, ab welcher Position (Spalte) innerhalb des Empfangsbereichs die empfangenen Daten in die Jobvariable geschrieben werden sollen. Bei POSITION != 1 wird zum Setzen der Jobvariable im Empfangsbereich auf die entsprechende Distanz positioniert.

VALUE = <c-string 1..256> / <x-string 1..512>
Die Jobvariable wird entsprechend der oben definierten Position und Länge mit dem hier übergebenen String besetzt.

Hinweis zur Verwendung von Jobvariablen:

Vor Ausführung der Anweisung wird die Jobvariable ab der angegebenen Position und in der angegebenen Länge mit Leerzeichen initialisiert. Tritt bei diesem Zugriff auf die Jobvariable ein Fehler auf, wird die Anweisung als ganzes nicht ausgeführt.

Beispiel

1. Es wird das Administrationskommando KDCSHUT WARN, TIME=01 aufgerufen.

   ```
   //SELECT-SERVICE SERVICE-NAME=KDCSHUT, SERVICE-DATA='WARN,TIME=01'
   ```

2. Es werden mit dem Kommando KDCINF die Eigenschaften der Benutzerkennung UPCUSER gelesen (KDCINF USER,LIST=UPCUSER). Die Ausgabe soll in die Jobvariable JV.USER geschrieben werden (ab Spalte 81).

   ```
   //SELECT-SERVICE SERVICE-NAME=KDCINF,                    -
   //               SERVICE-DATA='USER,LIST=(UPCUSER)',     -
   //               SET-SERVICE-JV=*YES (                   -
   //                      JV-ID=*JV-NAME  (                -
   //                         JV-NAME=JV.USER,              -
   //                         POSITION=81  )                -
   //                                   )                   -
   //                      VALUE=*RECEIVE-MSG(POSITION=161)
   ```

Ergebnis

Nach Ausführung der Anweisung ist dann die Jobvariable JV.USER ab Spalte 81 folgendermaßen belegt:

```
......Spalte 81                                  Spalte 123
............|                                          |
............UPCUSER0_____________OFF_____N_____________
............8________0_______0__UPCLT#T0_______.
............|                                  |
.....Spalte 124                        Spalte 160 ist next line X'15'
```

Bedeutung des Inhalts (siehe Abschnitt "Ausgabe von KDCINF (Beispiele)"(type=USER)):

UPCUSER (Wert von USER):
Name der Benutzerkennung.

OFF (Wert von STATUS):
Die Benutzerkennung ist gesperrt.

N (Wert von OSERV):
Die Benutzerkennung bearbeitet z.Zt. keinen Vorgang.

8 (Wert von NR.TACS):
Unter dieser Benutzerkennung wurden bisher 8 Transaktionsaufträge eingegeben.

0 (Wert von SECCNT):
Anzahl der Sicherheitsverletzungen unter dieser Benutzerkennung.

UPCLT#T0 (Wert von LTERM):
Name des LTERM-Partners, über den sich die Benutzerkennung anmeldet.

SHOW-CONFIGURATION

Mit SHOW-CONFIGURATION können Sie sich die Werte anzeigen lassen, die in der letzten CREATE-CONFIGURATION- oder MODIFY-CONFIGURATION-Anweisung festgelegt wurden.

SHOW-CONFIGURATION	Kurzname: **SHOWATTR**
CONFIGURATION-ID = **1** / <integer_1..1> ,**OUTPUT** = ***SYSOUT** /***LOGGING-FILE**	

CONFIGURATION-ID = 1

dient der Identifikation der Konfiguration. Es ist nur der Wert 1 zulässig.

OUTPUT=

gibt an, wo die angeforderten Daten ausgegeben werden sollen.

*SYSOUT

Die angeforderten Daten werden auf SYSOUT ausgegeben.

*LOGGING-FILE

Die angeforderten Daten sollen in die Protokolldatei geschrieben werden (siehe CREATE-CONFIGURATION; Operand WRITE-LOGGING-FILE). Wurde in CREATE-CONFIGURATION bzw. MODIFY-CONFIGURATION keine Protokolldatei definiert, dann wird die Ausgabe der Daten auf SYSOUT umgelenkt.

Beispiel

```
-> //SHOW-CONF OUTPUT=*SYSOUT
<- -- current configuration data: ---------------------------------
   local name = UPCPT#T0
   symbolic destination name = DBSADMT0
   partner name (from upicfile) = DBSUPAT0
   program name is enabled to UPIC
   no user identification given
   logging file name = LOG.CALLUTM
      open mode = replace
      record length =  79
      logging info = transmitted and received messages
      file is open
   upic trace is switched off
   program monitoring job variable is not specified
   encryption is not available
   -- end configuration data -------------------------------------
->
```

14.3.3 Bestandteile, Systemumgebung, Softwarekonfiguration auf dem BS2000-System

Für CALLUTM werden folgende Bestandteile ausgeliefert.

- Das Programm SYSPRG.UTM.070.CALLUTM
- Die SDF-Syntaxdatei SYSSDF.UTM.070.CALLUTM

Das Programm CALLUTM ist in der LMS-Bibliothek SYSLNK.UTM.070.CALLUTM enthalten. Es setzt folgende Softwarekonfiguration voraus:

- BS2000-System mit OSD/BC ab V10.0A
- CMX(BS2000) ab V1.4, falls zur Kommunikation CMX verwendet werden soll
- SDF ab V4.7C
- JV ab V15.0A (Jobvariablen)

Die Jobvariablen werden verwendet mit den Kettungsnamen UPICFIL, UPICPAT und UPICTRA wie im Handbuch „openUTM-Client für das Trägersystem UPIC“ beschrieben.

14.3.4 Integration in eine UTM-Anwendung auf dem BS2000-System

Damit das Programm CALLUTM mit einer UTM-Anwendung kommunizieren kann, sind die Einträge und Definitionen analog dem Beispiel im Abschnitt "Generierung" auf die aktuelle Anwendung zu übertragen bzw. zu modifizieren.

14.3.5 Programmüberwachende Jobvariable auf dem BS2000-System

Wird das Programm mit einer programmüberwachenden Jobvariablen gestartet (Operand MONJV beim Aufruf), dann wird diese Jobvariable zusätzlich zu den vom Betriebssystem gesetzten Werten von CALLUTM mit den folgenden Werten versorgt:

Spalte	Länge	Inhalt	Bedeutung
129	1	D / P / B	Modus, in dem CALLUTM abläuft. D: CALLUTM läuft im Dialog P: CALLUTM läuft im Dialog in einer Prozedur B: CALLUTM läuft in einem Batch-Auftrag
131 - 134	4	<tsn>	Task Sequence Number des Auftrags
136 - 139	4	<nnnn>	Laufende Nummer der Anweisung in diesem Programmlauf (SDF-Standardanweisungen werden nicht mitgezählt), führende Nullen werden unterdrückt.
141 - 148	8	CUA<name>	Interner Name der zuletzt ausgeführten oder der aktuellen Anweisung. Folgende Werte für <name> können auftreten: • CREA für CREATE-CONFIGURATION • MODC für MODIFY-CONFIGURATION • SHOWC für SHOW-CONFIGURATION • SELS für SELECT-SERVICE • CONTS für CONTINUE-SERVICE • DEALL für DEALLOCATE-CONVERSATION
150	1	C / N / O	Status der Service-Bearbeitung: • C: In der Server-Anwendung ist noch ein Service offen, er muss mit CONTINUE-SERVICE fortgesetzt werden. • N: Es ist kein Service mehr offen. • O: Ein Service wurde aufgerufen, es wurde jedoch noch keine Nachricht von ihm empfangen.
152 - 159	8	<servname>	Name des Service in der UTM-Anwendung (Transaktionscode)
161 - 168	8	<localnam>	„local name“: Aktueller Name, mit dem CALLUTM beim Trägersystem UPIC angemeldet ist.
170 - 177	8	<symbdest>	„symbolic destination name“ der UTM-Anwendung, mit der CALLUTM zur Zeit verbunden ist bzw. zu der CALLUTM eine Verbindung aufbaut (Name wie in der UPICFILE definiert).
179 - 210	32	<partner>	Name der UTM-Anwendung, der in der UPICFILE mit dem „symbolic destination name“ verknüpft ist.
251 - 256	6	<nnnnnn>	Fehlernummer, die zur Programmbeendigung führte, führende Nullen werden unterdrückt. 0 heißt normale Beendigung.

14.3.6 Meldungen von CALLUTM (BS2000-Systeme)

CALLUTM erzeugt folgende Meldungen:

`CUA0010: give attributes`

Diese Meldung wird nach dem Start von CALLUTM ausgegeben und fordert zur Eingabe der Anweisung CREATE-CONFIGURATION auf.

`CUA0015: symb-dest-name not found in UPICFILE`

Die Meldung kann nach dem Absetzen der Anweisungen CREATE-CONFIGURATION oder MODIFY-CONFIGURATION auftreten. Sie wird ausgegeben, wenn der im Operanden SYMB-DEST-NAME angegebene Name in der UPICFILE nicht existiert. Die Anweisung wird abgebrochen.
In Prozeduren führt dies zum Programmabbruch.

`CUA0020: conversation started by service <name> has to be continued`

Die Meldung zeigt an, dass ein zuvor mit SELECT-SERVICE gestarteter Service fortgesetzt werden muss. Sie können dann den Service mit CONTINUE-SERVICE fortsetzen oder mit DEALLOCATE-CONVERSATION abbrechen.
In <name> wird der Transaktionscode ausgegeben, mit dem der Service gestartet wurde.

`CUA0025: error analysing SDF statement = nnnn`

Bei der Analyse einer SDF-Anweisung ist ein Fehler aufgetreten. nnnn bezeichnet die Stelle im Programm, an der der Fehler aufgetreten ist.
In Prozeduren führt dies zum Programmabbruch.

`CUA0030: JV = <name> not accessible (error = nnnnnnn)`

Beim Initialisieren der Jobvariable <name> ist Fehler nnnnnnn aufgetreten. Das Programm versucht, den Fehler zu korrigieren. Gelingt dies nicht, führt das in Prozeduren zum Programmabbruch.

`callutm:error in <upic-call>:n`

`CUA0035: error in send-receive routine = nnnn`

Bei einem Aufruf an das Trägersystem UPIC ist ein Fehler aufgetreten. Vor der Ausgabe der Meldung CUA0035 informiert CALLUTM zunächst mit „`callutm:...`“ über den UPIC-Aufruf (<upic-call>), bei dem der Fehler aufgetreten ist, und den aufgetretenen UPIC-Returncode n (Bedeutung siehe Handbuch „openUTM-Client für das Trägersystem UPIC“). nnnn in CUA0035 bezeichnet die Stelle im Programm, an der der Fehler aufgetreten ist. In Prozeduren führt dieser Fehler zum Programmabbruch.

Beispiel

```
//**- callutm: error in allocate: 1 -**//

CUA0035: error in send-receive routine = 2007
```

Dies bedeutet, dass keine Verbindung zu der UTM-Anwendung aufgebaut werden konnte. Diese Meldung wird ausgegeben, wenn die UTM-Anwendung nicht verfügbar ist, die für die UTM-Anwendung abgesetzten BCMAP-Kommandos fehlerhaft waren oder für die UTM-Anwendung noch keine BCMAP-Kommandos abgesetzt wurden.

`CUA0040: not processed statement = <name>`

`CUA0045: program run is continued with next statement`

Die Anweisung <name> konnte nicht ausgeführt werden. Der Programmlauf wird nicht abgebrochen, Sie können ihn fortsetzen, indem Sie weitere Anweisungen absetzen. In Prozeduren wird der Programmlauf mit der folgenden Anweisung fortgesetzt.

`CUA0050: configuration modified`

Die Meldung kann nach dem Absetzen einer MODIFY-CONFIGURATION-Anweisung auftreten. Sie gibt an, dass die Anweisung erfolgreich ausgeführt wurde und die Konfiguration entsprechend der Angaben geändert wurde. Es kann mit der neuen Konfiguration weitergearbeitet werden.

`CUA0051: no value given to modify configuration`

Die Meldung kann nach dem Absetzen einer MODIFY-CONFIGURATION-Anweisung auftreten. Sie gibt an, dass die Anweisung erfolgreich analysiert wurde, die eingegebenen Werte aber zu keiner Änderung der Konfiguration geführt haben.

`CUA0055: conversation deallocated and abnormal end`

Die Conversation zur UTM-Anwendung wurde abgebaut. Der Service in der UTM-Anwendung wurde abgebrochen.
In Prozeduren führt dies zum Programmabbruch.

Mögliche Fehlerursachen:

- Der Service in der UTM-Anwendung lief auf einen Fehler (PEND ER).
- Es wurde ein Service zur Administration gestartet, für den keine Berechtigung seitens des Client besteht.

`CUA0060: no logging file assigned, output re-assigned to sysout (stdout)`

Die Meldung kann bei der SHOW-CONFIGURATION-Anweisung auftreten, bei der mit OUTPUT=*LOGGING-FILE die Ausgabe in eine Protokolldatei angefordert wurde. Die Meldung zeigt an, dass noch keine Protokolldatei zugewiesen ist, und die Ausgabe stattdessen auf SYSOUT erfolgt.

`CUA0065: deallocate not executed (program not in conversation),`

`program run can be continued with next statement"`

Die Anweisung DEALLOCATE-CONVERSATION wurde eingegeben, es war aber kein Vorgang offen.

Die Ausgabe erfolgt nach SYSOUT.

`CUA0070: restart not possible, continue with new service`

Ein Wiederanlauf mit KDCDISP war nicht möglich.

`CUA0080: for a list of c-strings the total c-string-length (1800) is exceeded`

In der Anweisung SELECT-SERVICE oder CONTINUE-SERVICE wurde im Operanden SERVICE-DATA eine Liste von C-Strings eingegeben. Die Länge der Daten überschreitet die maximal zulässige Länge.

`CUA0085: current conversation will be terminated`

CALLUTM läuft innerhalb einer Prozedur oder im Batch-Betrieb und vom Transportprotokoll UPIC wird ein Fehler gemeldet. CALLUTM beendet den offenen Vorgang und verzweigt gegebenenfalls zur Anweisung CALLUTM-ERROR-STEP oder erreicht das Ende der Anweisungen.

CUA0090: encryption is not available in this environment

Verschlüsselung nicht verfügbar.

Maßnahme: Verschlüsselung in die aktuelle UPIC-Client-Bibliothek aufnehmen.

CUA0100: CALLUTM-ERROR-STEP reached

Nachdem in einer Prozedur oder im Batchbetrieb ein Fehler auftrat, wurden alle Anweisungen übergangen bis CALLUTM-ERROR-STEP erkannt wurde.

CUA0105: all statements will be ignored until CALLUTM-ERROR-STEP is

recognized

Nachdem in einer Prozedur oder im Batchbetrieb ein Fehler auftrat, werden alle Anweisungen uebergangen, bis CALLUTM-ERROR-STEP gefunden wird. Wird keine solche Anweisung gefunden, beendet END den Programmlauf.

15 Fachwörter

Fachwörter, die an anderer Stelle erklärt werden, sind mit *kursiver* Schrift ausgezeichnet.

Ablaufinvariantes Programm
reentrant program

siehe *reentrant-fähiges Programm*.

Abnormale Beendigung einer UTM-Anwendung
abnormal termination of a UTM application

Beendigung einer *UTM-Anwendung*, bei der die *KDCFILE* nicht mehr aktualisiert wird. Eine abnormale Beendigung wird ausgelöst durch einen schwerwiegenden Fehler, z.B. Rechnerausfall, Fehler in der Systemsoftware. Wird die Anwendung erneut gestartet, führt openUTM einen *Warmstart* durch.

Abstrakte Syntax (OSI)
abstract syntax

Eine abstrakte Syntax ist die Menge der formal beschriebenen Datentypen, die zwischen Anwendungen über *OSI TP* ausgetauscht werden sollen. Eine abstrakte Syntax ist unabhängig von der eingesetzten Hardware und der jeweiligen Programmiersprache.

Access-List
access list

Eine Access-List definiert die Berechtigung für den Zugriff auf einen bestimmten *Service*, auf eine bestimmte *TAC-Queue* oder auf eine bestimmte *USER-Queue*. Eine Access-List ist als *Keyset* definiert und enthält einen oder mehrere *Keycodes*, die jeweils eine Rolle in der Anwendung repräsentieren. Benutzer, LTERMs oder (OSI-)LPAPs dürfen nur dann auf den Service oder die *TAC-Queue*/*USER-Queue* zugreifen, wenn ihnen die entsprechenden Rollen zugeteilt wurden, d.h. wenn ihr *Keyset* und die Access-List mindestens einen gemeinsamen *Keycode* enthalten.

Access Point (OSI)

siehe *Dienstzugriffspunkt.*

ACID-Eigenschaften
ACID properties

Abkürzende Bezeichnung für die grundlegenden Eigenschaften von *Transaktionen*: Atomicity, Consistency, Isolation und Durability.

Administration
administration

Verwaltung und Steuerung einer *UTM-Anwendung* durch einen *Administrator* oder ein *Administrationsprogramm*.

Administrations-Journal
administration journal

siehe *Cluster-Administrations-Journal*.

Administrationskommando
administration command

Kommandos, mit denen der *Administrator* einer *UTM-Anwendung* Administrationsfunktionen für diese Anwendung durchführt. Die Administrationskommandos sind als *Transaktionscodes* realisiert.

Administrationsprogramm
administration program

Teilprogramm, das Aufrufe der *Programmschnittstelle für die Administration* enthält. Dies kann das Standard-Administrationsprogramm *KDCADM* sein, das mit openUTM ausgeliefert wird, oder ein vom Anwender selbst erstelltes Programm.

Administrator
administrator

Benutzer mit Administrationsberechtigung.

AES

AES (Advanced Encryption Standard) ist der aktuelle symmetrische Verschlüsselungsstandard, festgelegt vom NIST (National Institute of Standards and Technology), basierend auf dem an der Universität Leuven (B) entwickelten Rijndael-Algorithmus. Wird das AES-Verfahren verwendet, dann erzeugt der UPIC-Client für jede Sitzung einen AES-Schlüssel.

Akzeptor (CPI-C)
acceptor

Die Kommunikationspartner einer *Conversation* werden *Initiator* und Akzeptor genannt. Der Akzeptor nimmt die vom Initiator eingeleitete Conversation mit Accept_Conversation entgegen.

Anmelde-Vorgang (KDCS)
sign-on service

Spezieller *Dialog-Vorgang*, bei dem die Anmeldung eines Benutzers an eine UTM-Anwendung durch *Teilprogramme* gesteuert wird.

Anschlussprogramm
linkage program

siehe *KDCROOT*.

Anwendungsinformation
application information

Sie stellt die Gesamtmenge der von der *UTM-Anwendung* benutzten Daten dar. Dabei handelt es sich um Speicherbereiche und Nachrichten der UTM-Anwendung, einschließlich der aktuell auf dem Bildschirm angezeigten Daten. Arbeitet die UTM-Anwendung koordiniert mit einem Datenbanksystem, so gehören die in der Datenbank gespeicherten Daten ebenfalls zur Anwendungsinformation.

Anwendungs-Kaltstart
application cold start

siehe *Kaltstart*.

Anwendungsprogramm
application program

Ein Anwendungsprogramm bildet den Hauptbestandteil einer *UTM-Anwendung*. Es besteht aus der Main Routine *KDCROOT* und den *Teilprogrammen*. Es bearbeitet alle Aufträge, die an eine *UTM-Anwendung* gerichtet werden.

Anwendungs-Warmstart
application warm start

siehe *Warmstart*.

Apache Axis

Apache Axis (Apache eXtensible Interaction System) ist eine SOAP-Engine zur Konstruktion von darauf basierenden Web Services und Client-Anwendungen. Es existiert eine Implementierung in C++ und Java.

Apache Tomcat

Apache Tomcat stellt eine Umgebung zur Ausführung von Java-Code auf Web-Servern bereit, die im Rahmen des Jakarta-Projekts der Apache Software Foundation entwickelt wird. Es handelt sich um einen in Java geschriebenen Servlet-Container, der mithilfe des JSP-Compilers Jasper auch JavaServer Pages in Servlets übersetzen und ausführen kann. Dazu kommt ein kompletter HTTP-Server.

Application Context (OSI)
application context

Der Application Context ist die Menge der Regeln, die für die Kommunikation zwischen zwei Anwendungen gelten sollen. Dazu gehören z.B. die *abstrakten Syntaxen* und die zugeordneten *Transfer-Syntaxen*.

Application Entity (OSI)
application entity

Eine Application Entity (AE) repräsentiert alle für die Kommunikation relevanten Aspekte einer realen Anwendung. Eine Application Entity wird durch einen global (d.h. weltweit) eindeutigen Namen identifiziert, den *Application Entity Title* (AET). Jede Application Entity repräsentiert genau einen *Application Process*. Ein Application Process kann mehrere Application Entities umfassen.

Application Entity Qualifier (OSI)
application entity qualifier

Bestandteil des *Application Entity Titles*. Der Application Entity Qualifier identifiziert einen *Dienstzugriffspunkt* innerhalb der Anwendung. Ein Application Entity Qualifier kann unterschiedlich aufgebaut sein. openUTM unterstützt den Typ "Zahl".

Application Entity Title (OSI)
application entity title

Ein Application Entity Title ist ein global (d.h. weltweit) eindeutiger Name für eine *Application Entity*. Er setzt sich zusammen aus dem *Application Process Title* des jeweiligen *Application Process* und dem *Application Entity Qualifier.*

Application Process (OSI)
application process

Der Application Process repräsentiert im *OSI-Referenzmodell* eine Anwendung. Er wird durch den *Application Process Title* global (d.h. weltweit) eindeutig identifiziert.

Application Process Title (OSI)
application process title

Gemäß der OSI-Norm dient der Application Process Title (APT) zur global (d.h. weltweit) eindeutigen Identifizierung von Anwendungen. Er kann unterschiedlich aufgebaut sein. openUTM unterstützt den Typ *Object Identifier*.

Application Service Element (OSI)
application service element

Ein Application Service Element (ASE) repräsentiert eine Funktionsgruppe der Anwendungsschicht (Schicht 7) des *OSI-Referenzmodells*.

Association (OSI)
association

Eine Association ist eine Kommunikationsbeziehung zwischen zwei *Application Entities*. Dem Begriff Association entspricht der *LU6.1*-Begriff *Session*.

Asynchron-Auftrag
queued job

Auftrag, der vom Auftraggeber zeitlich entkoppelt durchgeführt wird. Zur Bearbeitung von Asynchron-Aufträgen sind in openUTM *Message Queuing* Funktionen integriert, vgl. *UTM-gesteuerte Queue* und *Service-gesteuerte Queue*. Ein Asynchron-Auftrag wird durch die *Asynchron-Nachricht*, den Empfänger und ggf. den gewünschten Ausführungszeitpunkt beschrieben.
Ist der Empfänger ein Terminal, ein Drucker oder eine Transportsystem-Anwendung, so ist der Asynchron-Auftrag ein *Ausgabe-Auftrag*; ist der Empfänger ein Asynchron-Vorgang derselben oder einer fernen Anwendung, so handelt es sich um einen *Hintergrund-Auftrag*.
Asynchron-Aufträge können *zeitgesteuerte Aufträge* sein oder auch in einen *Auftrags-Komplex* integriert sein.

Asynchron-Conversation
asynchronous conversation

CPI-C-Conversation, bei der nur der *Initiator* senden darf. Für den *Akzeptor* muss in der *UTM-Anwendung* ein asynchroner Transaktionscode generiert sein.

Asynchron-Nachricht
asynchronous message

Asynchron-Nachrichten sind Nachrichten, die an eine *Message Queue* gerichtet sind. Sie werden von der lokalen *UTM-Anwendung* zunächst zwischengespeichert und dann unabhängig vom Auftraggeber weiter verarbeitet. Je nach Empfänger unterscheidet man folgende Typen von Asynchron-Nachrichten:

- Bei Asynchron-Nachrichten an eine *UTM-gesteuerte Queue* wird die Weiterverarbeitung komplett durch openUTM gesteuert. Zu diesem Typ gehören Nachrichten, die einen lokalen oder fernen *Asynchron-Vorgang* starten (vgl. auch *Hintergrund-Auftrag*) und Nachrichten, die zur Ausgabe an ein Terminal, einen Drucker oder eine Transportsystem-Anwendung geschickt werden (vgl. auch *Ausgabe-Auftrag*).

- Bei Asynchron-Nachrichten an eine *Service-gesteuerte Queue* wird die Weiterverarbeitung durch einen *Service* der Anwendung gesteuert. Zu diesem Typ gehören Nachrichten an eine *TAC-Queue*, Nachrichten an eine *USER-Queue* und Nachrichten an eine *Temporäre Queue*. Die User-Queue und die Temporäre Queue müssen dabei zur lokalen Anwendung gehören, die TAC-Queue kann sowohl in der lokalen als auch in einer fernen Anwendung liegen.

Asynchron-Programm
asynchronous program

Teilprogramm, das von einem *Hintergrund-Auftrag* gestartet wird.

Asynchron-Vorgang (KDCS)
asynchronous service

Vorgang , der einen *Hintergrund-Auftrag* bearbeitet. Die Verarbeitung erfolgt entkoppelt vom Auftraggeber. Ein Asynchron-Vorgang kann aus einem oder mehreren Teilprogrammen /Transaktionen bestehen. Er wird über einen asynchronen *Transaktionscode* gestartet.

Auftrag
job

Anforderung eines *Services* , der von einer *UTM-Anwendung* zur Verfügung gestellt wird, durch Angabe eines *Transaktionscodes* . Siehe auch: *Ausgabe-Auftrag* , *Dialog-Auftrag* , *Hintergrund-Auftrag* , *Auftrags-Komplex*.

Auftraggeber-Vorgang
job-submitting service

Ein Auftraggeber-Vorgang ist ein *Vorgang*, der zur Bearbeitung eines Auftrags einen Service von einer anderen Server-Anwendung (*Auftragnehmer-Vorgang*) anfordert.

Auftragnehmer-Vorgang
job-receiving service

Ein Auftragnehmer-Vorgang ist ein *Vorgang*, der von einem *Auftraggeber-Vorgang* einer anderen Server-Anwendung gestartet wird.

Auftrags-Komplex
job complex

Auftrags-Komplexe dienen dazu, *Asynchron-Aufträgen Quittungsaufträge* zuzuordnen. Ein Asynchron-Auftrag innerhalb eines Auftrags-Komplexes wird *Basis-Auftrag* genannt.

Ausgabe-Auftrag
queued output job

Ausgabeaufträge sind *Asynchron-Aufträge*, die die Aufgabe haben, eine Nachricht, z.B. ein Dokument, an einen Drucker, ein Terminal oder eine Transportsystem-Anwendung auszugeben. Ausgabeaufträge werden ausschließlich von UTM-Systemfunktionen bearbeitet, d.h. für die Bearbeitung müssen keine Teilprogramme erstellt werden.

Authentisierung
authentication

siehe *Zugangskontrolle*.

Autorisierung
authorization

siehe *Zugriffskontrolle*.

Axis

siehe *Apache Axis*.

Basis-Auftrag
basic job

Asynchron-Auftrag in einem *Auftrags-Komplex*.

Basisformat
basic format

Format, in das der Terminal-Benutzer alle Angaben eintragen kann, die notwendig sind, um einen Vorgang zu starten.

Basisname
filebase

Basisname der UTM-Anwendung.
Auf BS2000-Systemen ist Basisname das Präfix für die *KDCFILE*, die *Benutzerprotokoll-Datei* USLOG und die *System-Protokolldatei* SYSLOG.
Auf Unix-, Linux- und Windows-Systemen ist Basisname der Name des Verzeichnisses, unter dem die KDCFILE, die Benutzerprotokoll-Datei USLOG, die System-Protokolldatei SYSLOG und weitere Dateien der UTM-Anwendung abgelegt sind.

Basisname der Knoten-Anwendung
node filebase

Dateinamens-Präfix bzw. Verzeichnisname für die *KDCFILE*, *Benutzerprotokoll-Datei* und *Systemprotokoll-Datei* der *Knoten-Anwendung*.

Basisname der UTM-Cluster-Anwendung
cluster filebase

Dateinamens-Präfix bzw. Verzeichnisname für die *UTM-Cluster-Dateien*.

Benutzerausgang
user exit

Begriff ersetzt durch *Event-Exit.*

Benutzerkennung
user ID

Bezeichner für einen Benutzer, der in der *Konfiguration* der *UTM-Anwendung* festgelegt ist (optional mit Passwort zur *Zugangskontrolle*) und dem spezielle Zugriffsrechte (*Zugriffskontrolle*) zugeordnet sind. Ein Terminal-Benutzer muss bei der Anmeldung an die UTM-Anwendung diesen Bezeichner (und ggf. das zugeordnete Passwort) angeben. Auf BS2000-Systemen ist außerdem eine Zugangskontrolle über *Kerberos* möglich.
Für andere Clients ist die Angabe der Benutzerkennung optional, siehe auch *Verbindungs-Benutzerkennung*.
UTM-Anwendungen können auch ohne Benutzerkennungen generiert werden.

Benutzer-Protokolldatei
user log file

Datei oder Dateigeneration, in die der Benutzer mit dem KDCS-Aufruf LPUT Sätze variabler Länge schreibt. Jedem Satz werden die Daten aus dem KB-Kopf des *KDCS-Kommunikationsbereichs* vorangestellt. Die Benutzerprotokolldatei unterliegt der Transaktionssicherung von openUTM.

Berechtigungsprüfung
sign-on check

siehe *Zugangskontrolle*.

Beweissicherung (BS2000-Systeme)
audit

Im Betrieb einer *UTM-Anwendung* können zur Beweissicherung sicherheitsrelevante UTM-Ereignisse von *SAT* protokolliert werden.

Bildschirm-Wiederanlauf
screen restart

Wird ein *Dialog-Vorgang* unterbrochen, gibt openUTM beim *Vorgangswiederanlauf* die *Dialog-Nachricht* der letzten abgeschlossenen *Transaktion* erneut auf dem Bildschirm aus, sofern die letzte Transaktion eine Nachricht auf den Bildschirm ausgegeben hat.

Browsen von Asynchron-Nachrichten
browsing asynchronous messages

Ein *Vorgang* liest nacheinander die *Asynchron-Nachrichten*, die sich in einer *Service-gesteuerten Queue* befinden. Die Nachrichten werden während des Lesens nicht gesperrt und verbleiben nach dem Lesen in der Queue. Dadurch ist gleichzeitiges Lesen durch unterschiedliche Vorgänge möglich.

Bypass-Betrieb (BS2000-Systeme)
bypass mode

Betriebsart eines Druckers, der lokal an ein Terminal angeschlossen ist. Im Bypass-Betrieb wird eine an den Drucker gerichtete *Asynchron-Nachricht* an das Terminal gesendet und von diesem auf den Drucker umgeleitet, ohne auf dem Bildschirm angezeigt zu werden.

Cache-Speicher
cache

Pufferbereich zur Zwischenspeicherung von Anwenderdaten für alle Prozesse einer *UTM-Anwendung*. Der Cache-Speicher dient zur Optimierung der Zugriffe auf den *Pagepool* und für UTM-Cluster-Anwendungen zusätzlich auf den *Cluster-Pagepool*.

CCR (Commitment, Concurrency and Recovery)

CCR ist ein von OSI definiertes Application Service Element (ASE) für die OSI-TP-Kommunkation, welches die Protokollelemente (Services) zum Beginn und Abschluss (Commit oder Rollback) einer *Transaktion* enthält. CCR unterstützt das Zwei-Phasen-Commitment.

CCS-Name (BS2000-Systeme)
CCS name

siehe *Coded-Character-Set-Name* .

Client
client

Clients einer *UTM-Anwendung* können sein:

- Terminals
- UPIC-Client-Programme
- Transportsystem-Anwendungen (z.B. DCAM-, PDN-, CMX-, Socket-Anwendungen oder UTM-Anwendungen, die als *Transportsystem-Anwendung* generiert sind)

Clients werden über LTERM-Partner an die UTM-Anwendung angeschlossen.
Hinweis: UTM-Clients mit Trägersystem OpenCPIC werden wie *OSI TP-Partner* behandelt.

Client-Seite einer Conversation
client side of a conversation

Begriff ersetzt durch *Initiator*.

Cluster

Eine Anzahl von Rechnern, die über ein schnelles Netzwerk verbunden sind und die von außen in vielen Fällen als ein Rechner gesehen werden können. Das Ziel des "Clustering" ist meist die Erhöhung der Rechenkapazität oder der Verfügbarkeit gegenüber einem einzelnen Rechner.

Cluster-Administrations-Journal
cluster administration journal

Das Cluster-Administrations-Journal besteht aus:

- zwei Protokolldateien mit Endungen JRN1 und JRN2 für globale Administrationsaktionen,
- der JKAA-Datei, die eine Kopie der KDCS Application Area (KAA) enthält. Aus dieser Kopie werden administrative Änderungen übernommen, die nicht mehr in den beiden Protokolldateien enthalten sind.

Die Administrations-Journal-Dateien dienen dazu, administrative Aktionen, die in einer UTM-Cluster-Anwendung Cluster-weit auf alle Knoten-Anwendungen wirken sollen, an die anderen Knoten-Anwendungen weiterzugeben.

Cluster-GSSB-Datei
cluster GSSB file

Datei zur Verwaltung von GSSBs in einer *UTM-Cluster-Anwendung*. Die Cluster-GSSB-Datei wird mit dem UTM-Generierungstool *KDCDEF* erstellt.

Cluster-Konfigurationsdatei
cluster configuration file

Datei, die die zentralen Konfigurationsdaten einer *UTM-Cluster-Anwendung* enthält. Die Cluster-Konfigurationsdatei wird mit dem UTM-Generierungstool *KDCDEF* erstellt.

Cluster-Lock-Datei
cluster lock file

Datei einer *UTM-Cluster-Anwendung*, die dazu dient, Knoten-übergreifende Sperren auf Anwenderdatenbereiche zu verwalten.

Cluster-Pagepool
cluster pagepool

Der Cluster-Pagepool besteht aus einer Verwaltungsdatei und bis zu 10 Dateien, in denen die Cluster-weit verfügbaren Anwenderdaten (Vorgangsdaten inklusive LSSB, GSSB und ULS) einer *UTM-Cluster-Anwendung* gespeichert werden. Der Cluster-Pagepool wird mit dem UTM-Generierungstool *KDCDEF* erstellt.

Cluster-Startserialisierungs-Datei
cluster start serialization file

Lock-Datei, mit der die Starts einzelner Knoten-Anwendungen serialisiert werden (nur auf Unix-, Linux- und Windows-Systemen).

Cluster-ULS-Datei
cluster ULS file

Datei zur Verwaltung von ULS-Bereichen einer *UTM-Cluster-Anwendung*. Die Cluster-ULS-Datei wird mit dem UTM-Generierungstool *KDCDEF* erstellt.

Cluster-User-Datei
cluster user file

Datei, die die Verwaltungsdaten der Benutzer einer *UTM-Cluster-Anwendung* enthält. Die Cluster-User-Datei wird mit dem UTM-Generierungstool *KDCDEF* erstellt.

Coded-Character-Set-Name (BS2000-Systeme)
coded character set name

Bei Verwendung des Produkts *XHCS* (e**X**tended **H**ost **C**ode **S**upport) wird jeder verwendete Zeichensatz durch einen Coded-Character-Set-Namen (abgekürzt: "CCS-Name" oder "CCSN") eindeutig identifiziert.

Communication Resource Manager
communication resource manager

Communication Resource Manager (CRMs) kontrollieren in verteilten Systemen die Kommunikation zwischen den Anwendungsprogrammen. openUTM stellt CRMs für den internationalen Standard OSI TP, für den Industrie-Standard *LU6.1* und für das openUTM-eigene Protokoll UPIC zur Verfügung.

Contention Loser
contention loser

Jede Verbindung zwischen zwei Partnern wird von einem der Partner verwaltet. Der Partner, der die Verbindung verwaltet, heißt *Contention Winner*. Der andere Partner ist der Contention Loser.

Contention Winner
contention winner

Der Contention Winner einer Verbindung übernimmt die Verwaltung der Verbindung. Aufträge können sowohl vom Contention Winner als auch vom *Contention Loser* gestartet werden. Im Konfliktfall, wenn beide Kommunikationspartner gleichzeitig einen Auftrag starten wollen, wird die Verbindung vom Auftrag des Contention Winner belegt.

Conversation
conversation

Bei CPI-C nennt man die Kommunikation zwischen zwei CPI-C-Anwendungsprogrammen Conversation. Die Kommunikationspartner einer Conversation werden *Initiator* und *Akzeptor* genannt.

Conversation-ID
conversation ID

Jeder *Conversation* wird von CPI-C lokal eine Conversation-ID zugeordnet, d.h. *Initiator* und *Akzeptor* haben jeweils eine eigene Conversation-ID. Mit der Conversation-ID wird jeder CPI-C-Aufruf innerhalb eines Programms eindeutig einer Conversation zugeordnet.

CPI-C

CPI-C (**C**ommon **P**rogramming **I**nterface for **C**ommunication) ist eine von X/Open und dem CIW (**C**PI-C **I**mplementor's **W**orkshop) normierte Programmschnittstelle für die Programm-Programm-Kommunikation in offenen Netzen. Das in openUTM implementierte CPI-C genügt der CPI-C V2.0 CAE Specification von X/Open. Die Schnittstelle steht in COBOL und C zur Verfügung.
CPI-C in openUTM kann über die Protokolle OSI TP, LU6.1, UPIC und mit openUTM-LU6.2 kommunizieren.

Cross Coupled System / XCS

Verbund von BS2000-Rechnern mit *Highly Integrated System Complex* Multiple System Control Facility (HIPLEX® MSCF).

Datenraum (BS2000-Systeme)
data space

Virtueller Adressraum des BS2000, der in seiner gesamten Größe vom Anwender genutzt werden kann.
In einem Datenraum können nur Daten und als Daten abgelegte Programme adressiert werden, es kann kein Programmcode zum Ablauf gebracht werden.

Dead Letter Queue
dead letter queue

Die Dead Letter Queue ist eine *TAC-Queue* mit dem festen Namen KDCDLETQ. Sie steht immer zur Verfügung, um Asynchron-Nachrichten an *Transaktionscodes*, TAC-Queues, LPAP- oder OSI-LPAP-Partner zu sichern, die nicht verarbeitet werden konnten.
Die Sicherung von Asynchron-Nachrichten in der Dead Letter Queue kann durch den Parameter DEAD-LETTER-Q der TAC-, LPAP- oder OSI-LPAP-Anweisung für jedes Nachrichtenziel einzeln ein- und ausgeschaltet werden.

DES

DES (Data Encryption Standard) ist eine internationale Norm zur Verschlüsselung von Daten. Bei diesem Verfahren wird ein Schlüssel zum Ver- und Entschlüsseln verwendet. Wird das DES-Verfahren verwendet, dann erzeugt der UPIC-Client für jede Sitzung einen DES-Schlüssel.

Dialog-Auftrag
dialog job, interactive job

Auftrag, der einen *Dialog-Vorgang* startet. Der Auftrag kann von einem *Client* oder - bei *Server-Server-Kommunikation* - von einer anderen Anwendung erteilt werden.

Dialog-Conversation
dialog conversation

CPI-C-Conversation, bei der sowohl der *Initiator* als auch der *Akzeptor* senden darf. Für den *Akzeptor* muss in der *UTM-Anwendung* ein Dialog-Transaktionscode generiert sein.

Dialog-Nachricht
dialog message

Nachricht, die eine Antwort erfordert oder selbst eine Antwort auf eine Anfrage ist. Dabei bilden Anfrage und Antwort einen *Dialog-Schritt*.

Dialog-Programm
dialog program

Teilprogramm, das einen *Dialog-Schritt* teilweise oder vollständig bearbeitet.

Dialog-Schritt
dialog step

Ein Dialog-Schritt beginnt mit dem Empfang einer *Dialog-Nachricht* durch die *UTM-Anwendung*. Er endet mit der Antwort der UTM-Anwendung.

Dialog-Terminalprozess (Unix-, Linux- und Windows-Systeme)
dialog terminal process

Ein Dialog-Terminalprozess verbindet ein Unix-, Linux- oder Windows-Terminal mit den *Workprozessen* der *UTM-Anwendung*. Dialog-Terminalprozesse werden entweder vom Benutzer durch Eingabe von utmdtp oder über die LOGIN-Shell gestartet. Für jedes Terminal, das an eine UTM-Anwendung angeschlossen werden soll, ist ein eigener Dialog-Terminalprozess erforderlich.

Dialog-Vorgang
dialog service

Vorgang , der einen *Auftrag* im Dialog (zeitlich gekoppelt) mit dem Auftraggeber (*Client* oder eine andere Server-Anwendung) bearbeitet. Ein Dialog-Vorgang verarbeitet *Dialog-Nachrichten* vom Auftraggeber und erzeugt Dialog-Nachrichten für diesen. Ein Dialog-Vorgang besteht aus mindestens einer *Transaktion.* Ein Dialog-Vorgang umfasst in der Regel mindestens einen *Dialog-Schritt*. Ausnahme: Bei *Vorgangskettung* können auch mehrere Vorgänge einen Dialog-Schritt bilden.

Dienst
service

Programm auf Windows-Systemen, das im Hintergrund unabhängig von angemeldeten Benutzern oder Fenstern abläuft.

Dienstzugriffspunkt
service access point

Im *OSI-Referenzmodell* stehen einer Schicht am Dienstzugriffspunkt die Leistungen der darunterliegenden Schicht zur Verfügung. Der Dienstzugriffspunkt wird im lokalen System durch einen *Selektor* identifiziert. Bei der Kommunikation bindet sich die *UTM-Anwendung* an einen Dienstzugriffspunkt. Eine Verbindung wird zwischen zwei Dienstzugriffspunkten aufgebaut.

Distributed Transaction Processing

X/Open-Architekturmodell für die transaktionsorientierte *verteilte Verarbeitung*.

Druckadministration
print administration

Funktionen zur *Drucksteuerung* und Administration von *Ausgabeaufträgen*, die an einen Drucker gerichtet sind.

Druckerbündel
printer pool

Mehrere Drucker, die demselben *LTERM-Partner* zugeordnet sind.

Druckergruppe (Unix- und Linux-Systeme)
printer group

Die Unix- oder Linux-Plattform richtet für jeden Drucker standardmäßig eine Druckergruppe ein, die genau diesen Drucker enthält. Darüber hinaus lassen sich mehrere Drucker einer Druckergruppe, aber auch ein Drucker mehreren Druckergruppen zuordnen.

Druckerprozess (Unix- und Linux-Systeme)
printer process

Prozess, der vom *Mainprozess* zur Ausgabe von *Asynchron-Nachrichten* an eine *Druckergruppe* eingerichtet wird. Er existiert, solange die Druckergruppe an die *UTM-Anwendung* angeschlossen ist. Pro angeschlossener Druckergruppe gibt es einen Druckerprozess.

Druckersteuerstation
printer control terminal

Begriff wurde ersetzt durch *Druckersteuer-LTERM*.

Druckersteuer-LTERM
printer control LTERM

Über ein Druckersteuer-LTERM kann sich ein *Client* oder ein Terminal-Benutzer an eine *UTM-Anwendung* anschließen. Von dem Client-Programm oder Terminal aus kann dann die *Administration* der Drucker erfolgen, die dem Druckersteuer-LTERM zugeordnet sind. Hierfür ist keine Administrationsberechtigung notwendig.

Drucksteuerung
print control

openUTM-Funktionen zur Steuerung von Druckausgaben.

Dynamische Konfiguration
dynamic configuration

Änderung der *Konfiguration* durch die Administration. Im laufenden Betrieb der Anwendung können UTM-Objekte wie z.B. *Teilprogramme*, *Transaktionscodes*, *Clients*, *LU6.1-Verbindungen*, Drucker oder *Benutzerkennungen* in die Konfiguration aufgenommen, modifiziert oder teilweise auch gelöscht werden. Hierzu können die Administrationsprogramme WinAdmin oder WebAdmin verwendet werden, oder es müssen eigene *Administrationsprogramme* erstellt werden, die die Funktionen der *Programmschnittstelle der Administration* nutzen.

Einschritt-Transaktion
single-step transaction

Transaktion, die genau einen *Dialog-Schritt* umfasst.

Einschritt-Vorgang
single-step service

Dialog-Vorgang, der genau einen *Dialog-Schritt* umfasst.

Ereignisgesteuerter Vorgang
event-driven service

Begriff ersetzt durch *Event-Service*.

Event-Exit
event exit

Routine des *Anwendungsprogramms*, das bei bestimmten Ereignissen (z.B. Start eines Prozesses, Ende eines Vorgangs) automatisch gestartet wird. Diese darf - im Gegensatz zu den *Event-Services* - keine KDCS-, CPI-C- und XATMI-Aufrufe enthalten.

Event-Funktion
event function

Oberbegriff für *Event-Exits* und *Event-Services*.

Event-Service
event service

Vorgang, der beim Auftreten bestimmter Ereignisse gestartet wird, z.B. bei bestimmten UTM-Meldungen. Die *Teilprogramme* ereignisgesteuerter Vorgänge müssen KDCS-Aufrufe enthalten.

Funktionseinheit, Functional Unit (FU)
functional unit

Teilmenge des *OSI-TP*-Protokolls, die eine bestimmte Funktionalität beinhaltet. Das OSI-TP-Protokoll ist in folgende Funktionseinheiten aufgeteilt:

- Dialogue
- Shared Control
- Polarized Control
- Handshake
- Commit
- Chained Transactions
- Unchained Transactions
- Recovery

Ein Hersteller, der OSI-TP implementiert, muss nicht alle Funktionseinheiten realisieren, sondern kann sich auf eine Teilmenge beschränken. Eine Kommunikation zwischen Anwendungen zweier unterschiedlicher OSI-TP-Implementierungen ist nur dann möglich, wenn die realisierten Funktionseinheiten zueinander passen.

Generierung
generation

siehe *UTM-Generierung*.

Globaler Sekundärer Speicherbereich/GSSB
global secondary storage area

siehe *Sekundärspeicherbereich.*

Hardcopy-Betrieb
hardcopy mode

Betriebsart eines Druckers, der lokal an ein Terminal angeschlossen ist. Dabei wird eine Nachricht, die auf dem Bildschirm angezeigt wird, zusätzlich auf dem Drucker abgedruckt.

Heterogene Kopplung
heterogeneous link

Bei *Server-Server-Kommunikation:* Kopplung einer *UTM-Anwendung* mit einer Nicht-UTM-Anwendung, z.B. einer CICS- oder TUXEDO-Anwendung.

Highly Integrated System Complex / HIPLEX ®

Produktfamilie zur Realisierung eines Bedien-, Last- und Verfügbarkeitsverbunds mit mehreren BS2000-Servern.

Hintergrund-Auftrag
background job

Hintergrund-Aufträge sind *Asynchron-Aufträge*, die an einen *Asynchron-Vorgang* der eigenen oder einer fernen Anwendung gerichtet sind. Hintergrund-Aufträge eignen sich besonders für zeitintensive oder zeitunkritische Verarbeitungen, deren Ergebnis keinen direkten Einfluss auf den aktuellen Dialog hat.

HIPLEX ® MSCF

(MSCF = **M**ultiple **S**ystem **C**ontrol **F**acility)
stellt bei HIPLEX® die Infrastruktur sowie Basisfunktionen für verteilte Anwendungen bereit.

Homogene Kopplung
homogeneous link

Bei *Server-Server-Kommunikation*: Kopplung von *UTM-Anwendungen.* Dabei spielt es keine Rolle, ob die Anwendungen auf der gleichen oder auf unterschiedlichen Betriebssystem-Plattformen ablaufen.

Inbound-Conversation (CPI-C)
inbound conversation

siehe *Incoming-Conversation.*

Incoming-Conversation (CPI-C)
incoming conversation

Eine *Conversation*, bei der das lokale CPI-C-Programm *Akzeptor* ist, heißt Incoming-Conversation. In der X/Open-Specification wird für Incoming-Conversation auch das Synonym Inbound-Conversation verwendet.

Initiale KDCFILE
initial KDCFILE

In einer *UTM-Cluster-Anwendung* die *KDCFILE*, die von *KDCDEF* erzeugt wurde und vor dem Start der Knoten-Anwendungen für jeden Knoten kopiert werden muss.

Initiator (CPI-C)
initiator

Die Kommunikationspartner einer *Conversation* werden Initiator und *Akzeptor* genannt. Der Initiator baut die Conversation mit den CPI-C-Aufrufen Initialize_Conversation und Allocate auf.

Insert
insert

Feld in einem Meldungstext, in das openUTM aktuelle Werte einträgt.

Inverser KDCDEF
inverse KDCDEF

Funktion, die aus den Konfigurationsdaten der *KDCFILE*, die im laufenden Betrieb dynamisch angepasst wurde, Steueranweisungen für einen *KDCDEF*-Lauf erzeugt. Der inverse KDCDEF kann "offline" unter KDCDEF oder "online" über die *Programmschnittstelle zur Administration* gestartet werden.

IUTMDB
IUTMDB

Schnittstelle für die koordinierte Zusammenarbeit mit externen Resource Managern auf BS2000-Systemen. Dazu gehören Datenhaltungssysteme (LEASY) und Datenbanksysteme (SESAM/SQL, UDS/SQL).

JConnect-Client
JConnect client

Bezeichung für Clients auf Basis des Produkts openUTM-JConnect. Die Kommunikation mit der UTM-Anwendung erfolgt über das *UPIC-Protokoll*.

JDK

Java Development Kit
Standard-Entwicklungsumgebung von Oracle Corporation für die Entwicklung von Java-Anwendungen.

Kaltstart
cold start

Starten einer *UTM-Anwendung* nach einer *normalen Beendigung* der Anwendung oder nach einer Neugenerierung (vgl. auch *Warmstart*).

KDCADM

Standard-Administrationsprogramm, das zusammen mit openUTM ausgeliefert wird. KDCADM stellt Administrationsfunktionen zur Verfügung, die über Transaktionscodes (*Administrationskommandos*) aufgerufen werden.

KDCDEF

UTM-Tool für die *Generierung* von *UTM-Anwendungen*. KDCDEF erstellt anhand der Konfigurationsinformationen in den KDCDEF-Steueranweisungen die UTM-Objekte *KDCFILE* und die ROOT-Tabellen-Source für die Main Routine *KDCROOT*.
In UTM-Cluster-Anwendungen erstellt KDCDEF zusätzlich die *Cluster-Konfigurationsdatei,* die *Cluster-User-Datei*, den *Cluster-Pagepool*, die *Cluster-GSSB-Datei* und die *Cluster-ULS-Datei*.

KDCFILE

Eine oder mehrere Dateien, die für den Ablauf einer *UTM-Anwendung* notwendige Daten enthalten. Die KDCFILE wird mit dem UTM-Generierungstool *KDCDEF* erstellt. Die KDCFILE enthält unter anderem die *Konfiguration* der Anwendung.

KDCROOT

Main Routine eines *Anwendungsprogramms*, die das Bindeglied zwischen *Teilprogrammen* und UTM-Systemcode bildet. KDCROOT wird zusammen mit den *Teilprogrammen* zum *Anwendungsprogramm* gebunden.

KDCS-Parameterbereich
KDCS parameter area

siehe *Parameterbereich.*

KDCS-Programmschnittstelle
KDCS program interface

Universelle UTM-Programmschnittstelle, die den nationalen Standard DIN 66 265 erfüllt und Erweiterungen enthält. Mit KDCS (Kompatible Datenkommunikationsschnittstelle) lassen sich z.B. Dialog-Services erstellen und *Message Queuing* Funktionen nutzen. Außerdem stellt KDCS Aufrufe zur *verteilten Verarbeitung* zur Verfügung.

Kerberos

Kerberos ist ein standardisiertes Netzwerk-Authentisierungsprotokoll (RFC1510), das auf kryptographischen Verschlüsselungsverfahren basiert, wobei keine Passwörter im Klartext über das Netzwerk gesendet werden.

Kerberos-Principal
Kerberos principal

Eigentümer eines Schlüssels.
Kerberos arbeitet mit symmetrischer Verschlüsselung, d.h. alle Schlüssel liegen an zwei Stellen vor, beim Eigentümer eines Schlüssels (Principal) und beim KDC (Key Distribution Center).

Keycode
key code

Code, der in einer Anwendung eine bestimmte Zugriffsberechtigung oder eine bestimmte Rolle repräsentiert. Mehrere Keycodes werden zu einem *Keyset* zusammengefasst.

Keyset
key set

Zusammenfassung von einem oder mehrerer *Keycodes* unter einem bestimmten Namen. Ein Keyset definiert Berechtigungen im Rahmen des verwendeten Berechtigungskonzepts (Lock-/Keycode-Konzept oder *Access-List* -Konzept).
Ein Keyset kann einer *Benutzerkennung* , einem *LTERM-Partner* , einem *(OSI-) LPAP-Partner* , einem *Service* oder einer *TAC-Queue* zugeordnet werden.

Knoten
node

Einzelner Rechner eines *Clusters*.

Knoten-Anwendung
node application

UTM-Anwendung, die als Teil einer *UTM-Cluster-Anwendung* auf einem einzelnen *Knoten* zum Ablauf kommt.

Knoten-Recovery
node recovery

Wenn für eine abnormal beendete Knoten-Anwendung zeitnah kein Warmstart auf ihrem eigenen *Knoten-Rechner* möglich ist, kann man für diesen Knoten auf einem anderen Knoten des UTM-Clusters eine Knoten-Recovery (Wiederherstellung) durchführen. Dadurch können Sperren, die von der ausgefallenen Knoten-Anwendung gehalten werden, freigegeben werden, um die laufende *UTM-Cluster-Anwendung* nicht unnötig zu beeinträchtigen.

Knotengebundener Vorgang
node bound service

Ein knotengebundener Vorgang eines Benutzers kann nur an der Knoten-Anwendung fortgesetzt werden, an der der Benutzer zuletzt angemeldet war. Folgende Vorgänge sind immer knotengebunden:

- Vorgänge, die eine Kommunikation mit einem Auftragnehmer über LU6.1 oder OSI TP begonnen haben und bei denen der Auftragnehmervorgang noch nicht beendet wurde
- eingeschobene Vorgänge einer Vorgangskellerung
- Vorgänge, die eine SESAM-Transaktion abgeschlossen haben

Außerdem ist der Vorgang eines Benutzers knotengebunden, solange der Benutzer an einer Knoten-Anwendung angemeldet ist.

Kommunikationsbereich/KB (KDCS)
communication area

Transaktionsgesicherter KDCS-*Primärspeicherbereich*, der Vorgangs-spezifische Daten enthält. Der Kommunikationsbereich besteht aus 3 Teilen:

- dem KB-Kopf mit allgemeinen Vorgangsdaten
- dem KB-Rückgabebereich für Rückgaben nach KDCS-Aufrufen
- dem KB-Programmbereich zur Datenübergabe zwischen UTM-Teilprogrammen innerhalb eines *Vorgangs*.

Kommunikationsendpunkt
communication end point

siehe *Transportsystem-Endpunkt*

Konfiguration
configuration

Summe aller Eigenschaften einer *UTM-Anwendung*. Die Konfiguration beschreibt:

- Anwendungs- und Betriebsparameter
- die Objekte der Anwendung und die Eigenschaften dieser Objekte. Objekte sind z.B. *Teilprogramme* und *Transaktionscodes*, Kommunikationspartner, Drucker, *Benutzerkennungen*
- definierte Zugriffsschutz- und Zugangsschutzmaßnahmen

Die Konfiguration einer UTM-Anwendung wird bei der UTM-Generierung festgelegt (*statische Konfiguration*) und kann per *Administration* dynamisch (während des Anwendungslaufs) geändert werden (*dynamische Konfiguration*). Die Konfiguration ist in der *KDCFILE* abgelegt.

Logging-Prozess
logging process

Prozess auf Unix-, Linux- und Windows-Systemen, der die Protokollierung von Abrechnungssätzen oder Messdaten steuert.

Logische Verbindung
virtual connection

Zuordnung zweier Kommunikationspartner.

Log4j

Log4j ist ein Teil des Apache Jakarta Projekts. Log4j bietet Schnittstellen zum Protokollieren von Informationen (Ablauf-Informationen, Trace-Records,...) und zum Konfigurieren der Protokoll-Ausgabe. *WS4UTM* verwendet das Softwareprodukt Log4j für die Trace- und Logging-Funktionalität.

Lockcode

Code, um einen LTERM-Partner oder einen Transaktionscode vor unberechtigtem Zugriff zu schützen. Damit ist ein Zugriff nur möglich, wenn das *Keyset* des Zugreifenden den passenden *Keycode* enthält (Lock-/Keycode-Konzept).

Lokaler Sekundärer Speicherbereich/LSSB
local secondary storage area

siehe *Sekundärspeicherbereich.*

LPAP-Bündel
LPAP bundle

LPAP-Bündel ermöglichen die Verteilung von Nachrichten an LPAP-Partner auf mehrere Partner-Anwendungen. Soll eine UTM-Anwendung sehr viele Nachrichten mit einer Partner-Anwendung austauschen, kann es für die Lastverteilung sinnvoll sein, mehrere Instanzen der Partner-Anwendung zu starten und die Nachrichten auf die einzelnen Instanzen zu verteilen. In einem LPAP-Bündel übernimmt openUTM die Verteilung der Nachrichten an die Instanzen der Partner-Anwendung. Ein LPAP-Bündel besteht aus einem Master-LPAP und mehreren Slave-LPAPs. Die Slave-LPAPs werden dem Master-LPAP bei der UTM-Generierung zugeordnet. LPAP-Bündel gibt es sowohl für das OSI TP-Protokoll als auch für das LU6.1-Protokoll.

LPAP-Partner
LPAP partner

Für die *verteilte Verarbeitung* über das *LU6.1*-Protokoll muss in der lokalen Anwendung für jede Partner-Anwendung ein LPAP-Partner konfiguriert werden. Der LPAP-Partner spiegelt in der lokalen Anwendung die Partner-Anwendung wider. Bei der Kommunikation wird die Partner-Anwendung nicht über ihren Anwendungsnamen oder ihre Adresse, sondern über den Namen des zugeordneten LPAP-Partners angesprochen.

LTERM-Bündel
LTERM bundle

Ein LTERM-Bündel (Verbindungsbündel) besteht aus einem Master-LTERM und mehreren Slave-LTERMs. Mit einem LTERM-Bündel (Verbindungsbündel) verteilen Sie asynchrone Nachrichten an eine logische Partner-Anwendung gleichmäßig auf mehrere parallele Verbindungen.

LTERM-Gruppe
LTERM group

Eine LTERM-Gruppe besteht aus einem oder mehreren Alias-LTERMs, den Gruppen-LTERMs, und einem Primary-LTERM. In einer LTERM-Gruppe ordnen Sie mehrere LTERMs einer Verbindung zu.

LTERM-Partner
LTERM partner

Um *Clients* oder Drucker an eine *UTM-Anwendung* anschließen zu können, müssen in der Anwendung LTERM-Partner konfiguriert werden. Ein Client oder Drucker kann nur angeschlossen werden, wenn ihm ein LTERM-Partner mit entsprechenden Eigenschaften zugeordnet ist. Diese Zuordnung wird i.A. in der *Konfiguration* festgelegt, sie kann aber auch dynamisch über Terminal-Pools erfolgen.

LTERM-Pool
LTERM pool

Statt für jeden *Client* eine LTERM- und eine PTERM-Anweisung anzugeben, kann mit der Anweisung TPOOL ein Pool von LTERM-Partnern definiert werden. Schließt sich ein Client über einen LTERM-Pool an, wird ihm dynamisch ein LTERM-Partner aus dem Pool zugeordnet.

LU6.1

Geräteunabhängiges Datenaustauschprotokoll (Industrie-Standard) für die transaktionsgesicherte *Server-Server-Kommunikation*.

LU6.1-LPAP-Bündel
LU6.1-LPAP bundle

LPAP-Bündel für *LU6.1*-Partner-Anwendungen.

LU6.1-Partner
LU6.1 partner

Partner der *UTM-Anwendung*, der mit der UTM-Anwendung über das Protokoll *LU6.1* kommuniziert. Beispiele für solche Partner sind:

- eine UTM-Anwendung, die über LU6.1 kommuniziert
- eine Anwendung im IBM-Umfeld (z.B. CICS, IMS oder TXSeries), die über LU6.1 kommuniziert

Mainprozess (Unix-, Linux- und Windows-Systeme)
main process

Prozess, der die *UTM-Anwendung* startet. Er startet die *Workprozesse*, die *UTM-System-Prozesse*, *Druckerprozesse, Netzprozesse, Logging-Prozess* und den *Timerprozess* und überwacht die *UTM-Anwendung*.

Main Routine KDCROOT
main routine KDCROOT

siehe *KDCROOT*.

Management Unit
management unit

Komponente des *SE Servers*; ermöglicht mit Hilfe des *SE Managers* ein zentrales, web-basiertes Management aller Units eines SE Servers.

Meldung / UTM-Meldung
UTM message

Meldungen werden vom Transaktionsmonitor openUTM oder von UTM-Tools (wie z.B. *KDCDEF*) an *Meldungsziele* ausgegeben. Eine Meldung besteht aus einer Meldungsnummer und dem Meldungstext, der ggf. *Inserts* mit aktuellen Werten enthält. Je nach Meldungsziel werden entweder die gesamte Meldung oder nur Teile der Meldung (z.B. nur die Inserts) ausgegeben.

Meldungsdefinitionsdatei
message definition file

Die Meldungsdefinitionsdatei wird mit openUTM ausgeliefert und enthält standardmäßig die UTM-Meldungstexte in deutscher und englischer Sprache und die Definitionen der Meldungseigenschaften. Aufbauend auf diese Datei kann der Anwender auch eigene, individuelle Meldungsmodule erzeugen.

Meldungsziel
message destination

Ausgabemedium für eine *Meldung*. Mögliche Meldungsziele von Meldungen des Transaktionsmonitors openUTM sind z.B. Terminals, *TS-Anwendungen*, der *Event-Service* MSGTAC, die *System-Protokolldatei* SYSLOG oder *TAC-Queues, Asynchron-TACs, USER-Queues*, SYSOUT/SYSLST bzw. stderr/stdout. Meldungsziele von Meldungen der UTM-Tools sind SYSOUT /SYSLST bzw. stderr/stdout.

Mehrschritt-Transaktion
multi-step transaction

Transaktion, die aus mehr als einem *Verarbeitungsschritt* besteht.

Mehrschritt-Vorgang (KDCS)
multi-step service

Vorgang, der in mehreren *Dialog-Schritten* ausgeführt wird.

Message Queuing
message queuing

Message Queuing (MQ) ist eine Form der Kommunikation, bei der die Nachrichten (Messages) nicht unmittelbar, sondern über zwischengeschaltete *Message Queues* ausgetauscht werden. Sender und Empfänger können zeitlich und räumlich entkoppelt ablaufen. Die Übermittlung der Nachricht hängt nicht davon ab, ob gerade eine Netzverbindung besteht oder nicht. Bei openUTM gibt es *UTM-gesteuerte Queues* und *Service-gesteuerte Queues*.

Message Queue
message queue

Warteschlange, in der bestimmte Nachrichten transaktionsgesichert bis zur Weiterverarbeitung eingereiht werden. Je nachdem, wer die Weiterverarbeitung kontrolliert, unterscheidet man *Service-gesteuerte Queues* und *UTM-gesteuerte Queues*.

MSGTAC
MSGTAC

Spezieller Event-Service, der Meldungen mit dem Meldungsziel MSGTAC per Programm verarbeitet. MSGTAC ist ein Asynchron-Vorgang und wird vom Betreiber der Anwendung erstellt.

Multiplex-Verbindung (BS2000-Systeme)
multiplex connection

Spezielle Möglichkeit, die *OMNIS* bietet, um Terminals an eine *UTM-Anwendung* anzuschließen. Eine Multiplex-Verbindung ermöglicht es, dass sich mehrere Terminals eine *Transportverbindung* teilen.

Nachrichten-Bereich/NB (KDCS)
KDCS message area

Bei KDCS-Aufrufen: Puffer-Bereich, in dem Nachrichten oder Daten für openUTM oder für das *Teilprogramm* bereitgestellt werden.

Network File System/Service / NFS

Ermöglicht den Zugriff von Unix- und Linux-Rechnern auf Dateisysteme über das Netzwerk.

Netzprozess (Unix-, Linux- und Windows-Systeme)
net process

Prozess einer *UTM-Anwendung* zur Netzanbindung.

Netzwerk-Selektor
network selector

Der Netzwerk-Selektor identifiziert im lokalen System einen *Dienstzugriffspunkt* zur Vermittlungsschicht des *OSI-Referenzmodells*.

Normale Beendigung einer UTM-Anwendung
normal termination of a UTM application

Kontrollierte Beendigung einer *UTM-Anwendung*; das bedeutet u.a., dass die Verwaltungsdaten auf der *KDCFILE* aktualisiert werden. Eine normale Beendigung veranlasst der *Administrator* (z.B. mit KDCSHUT N). Den Start nach einer normalen Beendigung führt openUTM als *Kaltstart* durch.

Object Identifier
object identifier

Ein Object Identifier ist ein weltweit eindeutiger Bezeichner für Objekte im OSI-Umfeld. Ein Object Identifier besteht aus einer Folge von ganzen Zahlen, die einen Pfad in einer Baumstruktur repräsentiert.

Offener Terminalpool
open terminal pool

Terminalpool, der nicht auf *Clients* eines Rechners oder eines bestimmten Typs beschränkt ist. An diesen Terminalpool können sich alle Clients anschließen, für die kein Rechner- oder Typ-spezifischer Terminalpool generiert ist.

OMNIS (BS2000-Systeme)
OMNIS

OMNIS ist ein „Session-Manager“ auf einem BS2000-System, der die gleichzeitige Verbindungsaufnahme von einem Terminal zu mehreren Partnern in einem Netzwerk ermöglicht. OMNIS ermöglicht es außerdem, mit *Multiplex-Verbindungen* zu arbeiten.

Online-Import
online import

Als Online-Import wird in einer *UTM-Cluster-Anwendung* das Importieren von Anwendungsdaten aus einer normal beendeten Knoten-Anwendung in eine laufende Knoten-Anwendung bezeichnet.

Online-Update
online update

Als Online-Update wird in einer *UTM-Cluster-Anwendung* die Änderung der Konfiguration der Anwendung oder des Anwendungsprogramms oder der Einsatz einer neuen UTM-Korrekturstufe bei laufender *UTM-Cluster-Anwendung* bezeichnet.

OpenCPIC

Trägersystem für UTM-Clients, die das *OSI TP* Protokoll verwenden.

OpenCPIC-Client
OpenCPIC client

OSI TP Partner-Anwendungen mit Trägersystem *OpenCPIC*.

openSM2

Die Produktlinie openSM2 ist eine einheitliche Lösung für das unternehmensweite Performance Management von Server- und Speichersystemen. openSM2 bietet eine Messdatenerfassung, Online-Überwachung und Offline-Auswertung.

openUTM-Cluster
openUTM cluster

aus der Sicht von UPIC-Clients, **nicht** aus Server-Sicht:
Zusammenfassung mehrerer Knoten-Anwendungen einer UTM-Cluster-Anwendung zu einer logischen Anwendung, die über einen gemeinsamen Symbolic Destination Name adressiert wird.

openUTM-D

openUTM-D (openUTM-Distributed) ist eine openUTM-Komponente, die *verteilte Verarbeitung* ermöglicht. openUTM-D ist integraler Bestandteil von openUTM.

OSI-LPAP-Bündel
OSI-LPAP bundle

LPAP-Bündel für *OSI TP*-Partner-Anwendungen.

OSI-LPAP-Partner
OSI-LPAP partner

OSI-LPAP-Partner sind die bei openUTM generierten Adressen der *OSI TP-Partner*. Für die *verteilte Verarbeitung* über das Protokoll *OSI TP* muss in der lokalen Anwendung für jede Partner-Anwendung ein OSI-LPAP-Partner konfiguriert werden. Der OSI-LPAP-Partner spiegelt in der lokalen Anwendung die Partner-Anwendung wider. Bei der Kommunikation wird die Partner-Anwendung nicht über ihren Anwendungsnamen oder ihre Adresse, sondern über den Namen des zugeordneten OSI-LPAP-Partners angesprochen.

OSI-Referenzmodell
OSI reference model

Das OSI-Referenzmodell stellt einen Rahmen für die Standardisierung der Kommunikation von offenen Systemen dar. ISO, die Internationale Organisation für Standardisierung, hat dieses Modell im internationalen Standard
ISO IS7498 beschrieben. Das OSI-Referenzmodell unterteilt die für die Kommunikation von Systemen notwendigen Funktionen in sieben logische Schichten. Diese Schichten haben jeweils klar definierte Schnittstellen zu den benachbarten Schichten.

OSI TP

Von der ISO definiertes Kommunikationsprotokoll für die verteilte Transaktionsverarbeitung. OSI TP steht für Open System Interconnection Transaction Processing.

OSI TP-Partner
OSI TP partner

Partner der UTM-Anwendung, der mit der UTM-Anwendung über das OSI TP-Protokoll kommuniziert.
Beispiele für solche Partner sind:

- eine UTM-Anwendung, die über OSI TP kommuniziert
- eine Anwendung im IBM-Umfeld (z.B. CICS), die über openUTM-LU62 angeschlossen ist
- ein *OpenCPIC-Client*
- Anwendungen anderer TP-Monitore, die OSI TP unterstützen

Outbound-Conversation (CPI-C)
outbound conversation

siehe *Outgoing-Conversation*.

Outgoing-Conversation (CPI-C)
outgoing conversation

Eine Conversation, bei der das lokale CPI-C-Programm der *Initiator* ist, heißt Outgoing-Conversation. In der X/Open-Specification wird für Outgoing-Conversation auch das Synonym Outbound-Conversation verwendet.

Pagepool
page pool

Teil der *KDCFILE*, in dem Anwenderdaten gespeichert werden.
In einer *stand-alone Anwendung* sind dies z.B. *Dialog-Nachrichten*, Nachrichten an *Message Queues*, *Sekundärspeicherbereiche*.
In einer *UTM-Cluster-Anwendung* sind dies z.B. Nachrichten an *Message Queues, TLS*.

Parameterbereich
parameter area

Datenstruktur, in der ein *Teilprogramm* bei einem UTM-Aufruf die für diesen Aufruf notwendigen Operanden an openUTM übergibt.

Partner-Anwendung
partner application

Partner einer UTM-Anwendung bei *verteilter Verarbeitung*. Für die verteilte Verarbeitung werden höhere Kommunikationsprotokolle verwendet (*LU6.1*, *OSI TP* oder LU6.2 über das Gateway openUTM-LU62).

Postselection (BS2000-Systeme)
postselection

Auswahl der protokollierten UTM-Ereignisse aus der SAT-Protokolldatei, die ausgewertet werden sollen. Die Auswahl erfolgt mit Hilfe des Tools SATUT.

Programmraum (BS2000-Systeme)
program space

In Speicherklassen aufgeteilter virtueller Adressraum des BS2000, in dem sowohl ablauffähige Programme als auch reine Daten adressiert werden.

Prepare to commit (PTC)
prepare to commit

Bestimmter Zustand einer verteilten Transaktion:
Das Transaktionsende der verteilten Transaktion wurde eingeleitet, es wird jedoch noch auf die Bestätigung des Transaktionsendes durch den Partner gewartet.

Preselection (BS2000-Systeme)
preselection

Festlegung der für die *SAT-Beweissicherung* zu protokollierenden UTM-Ereignisse. Die Preselection erfolgt durch die UTM-SAT-Administration. Man unterscheidet Ereignis-spezifische, Benutzer-spezifische und Auftrags-(TAC-)spezifische Preselection.

Presentation-Selektor
presentation selector

Der Presentation-Selektor identifiziert im lokalen System einen *Dienstzugriffspunkt* zur Darstellungsschicht des *OSI-Referenzmodells*.

Primärspeicherbereich
primary storage area

Bereich im Arbeitsspeicher, auf den das *KDCS-Teilprogramm* direkt zugreifen kann, z.B. *Standard Primärer Arbeitsbereich*, *Kommunikationsbereich*.

Printerprozess (Unix- und Linux-Systeme)
printer process

siehe *Druckerprozess*.

Programmschnittstelle zur Administration
program interface for administration

UTM-Programmschnittstelle, mit deren Hilfe der Anwender eigene *Administrationsprogramme* erstellen kann. Die Programmschnittstelle zur Administration bietet u.a. Funktionen zur *dynamischen Konfiguration*, zur Modifikation von Eigenschaften und Anwendungsparametern und zur Abfrage von Informationen zur *Konfiguration* und zur aktuellen Auslastung der Anwendung.

Prozess
prozess

In den openUTM-Handbüchern wird der Begriff "Prozess" als Oberbegriff für Prozess (Unix-, Linux- und Windows-Systeme) und Task (BS2000-Systeme) verwendet.

Queue
queue

siehe *Message Queue*

Quick Start Kit

Beispielanwendung, die mit openUTM (Windows-Systeme) ausgeliefert wird.

Quittungs-Auftrag
confirmation job

Bestandteil eines *Auftrags-Komplexes*, worin der Quittungs-Auftrag dem *Basis-Auftrag* zugeordnet ist. Es gibt positive und negative Quittungsaufträge. Bei positivem Ergebnis des *Basis-Auftrags* wird der positive Quittungs-Auftrag wirksam, sonst der negative.

Redelivery
redelivery

Erneutes Zustellen einer *Asynchron-Nachricht*, nachdem diese nicht ordnungsgemäß verarbeitet werden konnte, z.B. weil die *Transaktion* zurückgesetzt oder der *Asynchron-Vorgang* abnormal beendet wurde. Die Nachricht wird wieder in die Message Queue eingereiht und lässt sich damit erneut lesen und/oder verarbeiten.

Reentrant-fähiges Programm
reentrant program

Programm, dessen Code durch die Ausführung nicht verändert wird.
Auf BS2000-Systemen ist dies Voraussetzung dafür, *Shared Code* zu nutzen.

Request
request

Anforderung einer *Service-Funktion* durch einen *Client* oder einen anderen Server.

Requestor
requestor

In XATMI steht der Begriff Requestor für eine Anwendung, die einen Service aufruft.

Resource Manager
resource manager

Resource Manager (RMs) verwalten Datenressourcen. Ein Beispiel für RMs sind Datenbank-Systeme. openUTM stellt aber auch selbst Resource Manager zur Verfügung, z.B. für den Zugriff auf *Message Queues*, lokale Speicherbereiche und Logging-Dateien. Anwendungsprogramme greifen auf RMs über RM-spezifische Schnittstellen zu. Für Datenbank-Systeme ist dies meist SQL, für die openUTM-RMs die Schnittstelle KDCS.

RFC1006

Von IETF (Internet Engineering Task Force) definiertes Protokoll der TCP/IP-Familie zur Realisierung der ISO-Transportdienste (Transportklasse 0) auf TCP/IP-Basis.

RSA

Abkürzung für die Erfinder des RSA-Verschlüsselungsverfahrens Rivest, Shamir und Adleman. Bei diesem Verfahren wird ein Schlüsselpaar verwendet, das aus einem öffentlichen und einem privaten Schlüssel besteht. Eine Nachricht wird mit dem öffentlichen Schlüssel verschlüsselt und kann nur mit dem privaten Schlüssel entschlüsselt werden. Das RSA-Schlüsselpaar wird von der UTM-Anwendung erzeugt.

SAT-Beweissicherung (BS2000-Systeme)
SAT audit

Beweissicherung durch die Komponente SAT (Security Audit Trail) des BS2000-Softwareproduktes SECOS.

SE Manager
SE manager

Web-basierte Benutzeroberfläche (GUI) für Business Server der SE Serie. Der SE Manager läuft auf der *Management Unit* und ermöglicht die zentrale Bedienung und Verwaltung von Server Units (mit /390-Architektur und/oder x86-Architektur), Application Units (x86-Architektur), Net Unit und der Peripherie.

SE Server
SE server

Ein Business Server der SE Serie von Fujitsu.

Sekundärspeicherbereich
secondary storage area

Transaktionsgesicherter Speicherbereich, auf den das KDCS-*Teilprogramm* mit speziellen Aufrufen zugreifen kann. Lokale Sekundärspeicherbereiche (LSSB) sind einem *Vorgang* zugeordnet, auf globale Sekundärspeicherbereiche (GSSB) kann von allen Vorgängen einer *UTM-Anwendung* zugegriffen werden. Weitere Sekundärspeicherbereiche sind der *Terminal-spezifische Langzeitspeicher (TLS)* und der *User-spezifische Langzeitspeicher (ULS)* .

Selektor
selector

Ein Selektor identifiziert im lokalen System einen *Zugriffspunkt* auf die Dienste einer Schicht des *OSI-Referenzmodells*. Jeder Selektor ist Bestandteil der Adresse des Zugriffspunktes.

Semaphor (Unix-, Linux- und Windows-Systeme)
semaphore

Betriebsmittel auf Unix-, Linux- und Windows-Systemen, das zur Steuerung und Synchronisation von Prozessen dient.

Server
server

Ein Server ist eine *Anwendung*, die *Services* zur Verfügung stellt. Oft bezeichnet man auch den Rechner, auf dem Anwendungen laufen, als Server.

Server-Seite einer Conversation (CPI-C)
server side of a conversation

Begriff ersetzt durch *Akzeptor*.

Server-Server-Kommunikation
server-server communication

siehe *verteilte Verarbeitung*.

Service Access Point

siehe *Dienstzugriffspunkt.*

Service
service

Services bearbeiten die Aufträge, die an eine Server-Anwendung geschickt werden. Ein Service in einer UTM-Anwendung wird auch Vorgang genannt und setzt sich aus einer oder mehreren Transaktionen zusammen. Ein Service wird über den Vorgangs-TAC aufgerufen. Services können von Clients oder anderen Services angefordert werden.

Service-gesteuerte Queue
service controlled queue

Message Queue, bei der der Abruf und die Weiterverarbeitung der Nachrichten durch Services gesteuert werden. Ein Service muss zum Lesen der Nachricht explizit einen KDCS-Aufruf (DGET) absetzen.
Service-gesteuerte Queues gibt es bei openUTM in den Varianten USER-Queue, TAC-Queue und Temporäre Queue.

Service Routine
service routine

siehe Teilprogramm.

Session
session

Kommunikationsbeziehung zweier adressierbarer Einheiten im Netz über das SNA-Protokoll LU6.1.

Session-Selektor
session selector

Der Session-Selektor identifiziert im lokalen System einen Zugriffspunkt zu den Diensten der Kommunikationssteuerschicht (Session-Layer) des OSI-Referenzmodells.

Shared Code (BS2000-Systeme)
shared code

Code, der von mehreren Prozessen gemeinsam benutzt werden kann.

Shared Memory
shared memory

Virtueller Speicherbereich, auf den mehrere Prozesse gleichzeitig zugreifen können.

Shared Objects (Unix-, Linux- und Windows-Systeme)
shared objects

Teile des Anwendungsprogramms können als Shared Objects erzeugt werden. Diese werden dynamisch zur Anwendung dazugebunden und können im laufenden Betrieb ausgetauscht werden. Shared Objects werden mit der KDCDEF-Anweisung SHARED-OBJECT definiert.

Sicherungspunkt
synchronization point, consistency point

Ende einer Transaktion. Zu diesem Zeitpunkt werden alle in der Transaktion vorgenommenen Änderungen der Anwendungsinformation gegen Systemausfall gesichert und für andere sichtbar gemacht. Während der Transaktion gesetzte Sperren werden wieder aufgehoben.

Single System Image

Unter single system image versteht man die Eigenschaft eines Clusters, nach außen hin als ein einziges, in sich geschlossenes System zu erscheinen. Die heterogene Natur des Clusters und die interne Verteilung der Ressourcen im Cluster ist für die Benutzer des Clusters und die Anwendungen, die mit dem Cluster kommunizieren, nicht sichtbar.

SOA

SOA (Service-oriented architecture).
SOA ist ein Konzept für eine Systemarchitektur, in dem Funktionen in Form von wieder verwendbaren, technisch voneinander unabhängigen und fachlich lose gekoppelten Services implementiert werden. Services können unabhängig von zugrunde liegenden Implementierungen über Schnittstellen aufgerufen werden, deren Spezifikationen öffentlich und damit vertrauenswürdig sein können. Service-Interaktion findet über eine dafür vorgesehene Kommunikationsinfrastruktur statt.

SOAP

SOAP (Simple Object Access Protocol) ist ein Protokoll, mit dessen Hilfe Daten zwischen Systemen ausgetauscht und Remote Procedure Calls durchgeführt werden können. SOAP stützt sich auf die Dienste anderer Standards, XML zur Repräsentation der Daten und Internet-Protokolle der Transport- und Anwendungsschicht zur Übertragung der Nachrichten.

Socket-Verbindung
socket connection

Transportsystem-Verbindung, die die Socket-Schnittstelle verwendet. Die Socket-Schnittstelle ist eine Standard-Programmschnittstelle für die Kommunikation über TCP/IP.

Stand-alone Anwendung
stand-alone application

siehe stand-alone UTM-Anwendung.

Stand-alone UTM-Anwendung
stand-alone UTM application

Herkömmliche UTM-Anwendung, die nicht Bestandteil einer UTM-Cluster-Anwendung ist.

Standard Primärer Arbeitsbereich/SPAB (KDCS)
standard primary working area

Bereich im Arbeitsspeicher, der jedem KDCS-Teilprogramm zur Verfügung steht. Sein Inhalt ist zu Beginn des Teilprogrammlaufs undefiniert oder mit einem Füllzeichen vorbelegt.

Startformat
start format

Format, das openUTM am Terminal ausgibt, wenn sich ein Benutzer erfolgreich bei der UTM-Anwendung angemeldet hat (ausgenommen nach Vorgangs-Wiederanlauf und beim Anmelden über Anmelde-Vorgang).

Statische Konfiguration
static configuration

Festlegen der Konfiguration bei der UTM-Generierung mit Hilfe des UTM-Tools KDCDEF.

SYSLOG-Datei
SYSLOG file

siehe System-Protokolldatei.

System-Protokolldatei
system log file

Datei oder Dateigeneration, in die openUTM während des Laufs einer UTM-Anwendung alle UTM-Meldungen protokolliert, für die das Meldungsziel SYSLOG definiert ist.

TAC
TAC

siehe Transaktionscode.

TAC-Queue
TAC queue

Message Queue, die explizit per KDCDEF-Anweisung generiert wird. Eine TAC-Queue ist eine Service-gesteuerte Queue und kann unter dem generierten Namen von jedem Service aus angesprochen werden.

Teilprogramm
program unit

UTM-Services werden durch ein oder mehrere Teilprogramme realisiert. Die Teilprogramme sind Bestandteile des Anwendungsprogramms. Abhängig vom verwendeten API müssen sie KDCS-, XATMI- oder CPIC-Aufrufe enthalten. Sie sind über Transaktionscodes ansprechbar. Einem Teilprogramm können mehrere Transaktionscodes zugeordnet werden.

Temporäre Queue
temporary queue

Message Queue, die dynamisch per Programm erzeugt wird und auch wieder per Programm gelöscht werden kann, vgl. Service-gesteuerte Queue.

Terminal-spezifischer Langzeitspeicher/TLS (KDCS)
terminal-specific long-term storage

Sekundärspeicher, der einem LTERM-, LPAP- oder OSI-LPAP-Partner zugeordnet ist und über das Anwendungsende hinaus erhalten bleibt.

Timerprozess (Unix-, Linux- und Windows-Systeme)
timer process

Prozess, der Aufträge zur Zeitüberwachung von Workprozessen entgegennimmt, sie in ein Auftragsbuch einordnet und nach einer im Auftragsbuch festgelegten Zeit den Workprozessen zur Bearbeitung wieder zustellt.

TLS Termination Proxy
TLS termination proxy

Ein TLS-Terminierungsproxy ist ein Proxy-Server, der verwendet wird, um eingehende TLS-Verbindungen zu verarbeiten, die Daten zu entschlüsseln und die unverschlüsselte Anforderung an andere Server weiterzugeben.

TNS (Unix-, Linux- und Windows-Systeme)

Abkürzung für den Transport Name Service, der einem Anwendungsnamen einen Transport-Selektor und das Transportsystem zuordnet, über das die Anwendung erreichbar ist.

Tomcat

siehe Apache Tomcat

Transaktion
transaction

Verarbeitungsabschnitt innerhalb eines Services, für den die Einhaltung der ACID-Eigenschaften garantiert wird. Von den in einer Transaktion beabsichtigten Änderungen der Anwendungsinformation werden entweder alle konsistent durchgeführt oder es wird keine durchgeführt (Alles-oder-Nichts Regel). Das Transaktionsende bildet einen Sicherungspunkt.

Transaktionscode/TAC
transaction code

Name, über den ein Teilprogramm aufgerufen werden kann. Der Transaktionscode wird dem Teilprogramm bei der statischen oder dynamischen Konfiguration zugeordnet. Einem Teilprogramm können auch mehrere Transaktionscodes zugeordnet werden.

Transaktionsrate
transaction rate

Anzahl der erfolgreich beendeten Transaktionen pro Zeiteinheit.

Transfer-Syntax
transfer syntax

Bei OSI TP werden die Daten zur Übertragung zwischen zwei Rechnersystemen von der lokalen Darstellung in die Transfer-Syntax umgewandelt. Die Transfer-Syntax beschreibt die Daten in einem neutralen Format, das von allen beteiligten Partnern verstanden wird. Jeder Transfer-Syntax muss ein Object Identifier zugeordnet sein.

Transport Layer Security
transport layer security

Der Transport Layer Security, ist ein hybrides Verschlüsselungsprotokoll zur sicheren Datenübertragung im Internet.

Transport-Selektor
transport selector

Der Transport-Selektor identifiziert im lokalen System einen Dienstzugriffspunkt zur Transportschicht des OSI-Referenzmodells.

Transportsystem-Anwendung
transport system application

Anwendung, die direkt auf einer Transportsystem-Schnittstelle wie z.B. CMX, DCAM oder Socket aufsetzt. Für den Anschluss von Transportsystem-Anwendungen muss bei der Konfiguration als Partnertyp APPLI oder SOCKET angegeben werden. Eine Transportsystem-Anwendung kann nicht in eine Verteilte Transaktion eingebunden werden.

Transportsystem-Endpunkt
transport system end point

Bei der Client-/Server- oder Server-/Server-Kommunikation wird eine Verbindung zwischen zwei Transportsystem-Endpunkten aufgebaut. Ein Transportsystem-Endpunkt wird auch als lokaler Anwendungsname bezeichnet und wird mit der Anweisung BCAMAPPL oder mit MAX APPLINAME definiert.

Transportsystem-Zugriffspunkt
transport system access point

siehe Transportsystem-Endpunkt.

Transportverbindung
transport connection

Im OSI-Referenzmodell eine Verbindung zwischen zwei Instanzen der Schicht 4 (Transportschicht).

TS-Anwendung
TS application

siehe Transportsystem-Anwendung.

Typisierter Puffer (XATMI)
typed buffer

Puffer für den Austausch von typisierten und strukturierten Daten zwischen Kommunikationspartnern. Durch diese typisierten Puffer ist die Struktur der ausgetauschten Daten den Partnern implizit bekannt.

UPIC

Trägersystem für UTM-Clients. UPIC steht für Universal Programming Interface for Communication. Die Kommunikation mit der UTM-Anwendung erfolgt über das UPIC-Protokoll.

UPIC-Client

Bezeichnung für UTM-Clients mit Trägersystem UPIC und JConnect-Clients.

UPIC-Protokoll
Upic protocol

Protokoll für die Client-Server-Kommunikation mit UTM-Anwendungen. Das UPIC-Protokoll wird von UPIC-Clients und von JConnect-Clients verwendet.

UPIC Analyzer

Komponente zur Analyse der mit UPIC Capture mitgeschnittenen UPIC-Kommunikation. Dieser Schritt dient dazu, den Mitschnitt für das Abspielen mit UPIC Replay aufzubereiten.

UPIC Capture

Mitschneiden der Kommunikation zwischen UPIC-Clients und UTM-Anwendungen, um sie zu einem späteren Zeitpunkt abspielen zu können
(UPIC Replay).

UPIC Replay

Komponente zum Abspielen der mit UPIC Capture mitgeschnittenen und mit UPIC Analyzer aufbereiteten UPIC-Kommunikation.

USER-Queue
USER queue

Message Queue, die openUTM jeder Benutzerkennung zur Verfügung stellt. Eine USER-Queue zählt zu den Service-gesteuerten Queues und ist immer der jeweiligen Benutzerkennung zugeordnet. Der Zugriff von fremden UTM-Benutzern auf die eigene USER-Queue kann eingeschränkt werden.

User-spezifischer Langzeitspeicher/ULS
user-specific long-term storage

Sekundärspeicher, der einer Benutzerkennung, einer Session oder einer Association zugeordnet ist und über das Anwendungsende hinaus erhalten bleibt.

USLOG-Datei
USLOG file

siehe Benutzer-Protokolldatei.

UTM-Anwendung
UTM application

Eine UTM-Anwendung stellt Services zur Verfügung, die Aufträge von Clients oder anderen Anwendungen bearbeiten. openUTM übernimmt dabei u.a. die Transaktionssicherung und das Management der Kommunikations- und Systemressourcen. Technisch gesehen ist eine UTM-Anwendung eine Prozessgruppe, die zur Laufzeit eine logische Server-Einheit bildet.

UTM-Client
UTM client

siehe Client.

UTM-Cluster-Anwendung
UTM cluster application

UTM-Anwendung, die für den Einsatz in einem Cluster generiert ist und die man logisch als **eine** Anwendung betrachten kann.
Physikalisch gesehen besteht eine UTM-Cluster-Anwendung aus mehreren, identisch generierten UTM-Anwendungen, die auf den einzelnen Knoten laufen.

UTM-Cluster-Dateien
UTM cluster files

Oberbegriff für alle Dateien, die für den Ablauf einer UTM-Cluster-Anwendung auf Unix-, Linux- und Windows-Systemen benötigt werden. Dazu gehören folgende Dateien:

- Cluster-Konfigurationsdatei
- Cluster-User-Datei
- Dateien des Cluster-Pagepool
- Cluster-GSSB-Datei
- Cluster-ULS-Datei
- Dateien des Cluster-Administrations-Journals*
- Cluster-Lock-Datei*
- Lock-Datei zur Start-Serialisierung*

Die mit * gekennzeichneten Dateien werden beim Start der ersten Knoten-Anwendung angelegt, alle anderen Dateien werden bei der Generierung mit KDCDEF erzeugt.

UTM-D

siehe openUTM-D.

UTM-Datenstation
UTM terminal

Begriff ersetzt durch LTERM-Partner.

UTM-F

UTM-Anwendungen können als UTM-F-Anwendungen (UTM-Fast) generiert werden. Bei UTM-F wird zugunsten der Performance auf Platteneingaben/-ausgaben verzichtet, mit denen bei UTM-S die Sicherung von Benutzer- und Transaktionsdaten durchgeführt wird. Gesichert werden lediglich Änderungen der Verwaltungsdaten.
In UTM-Cluster-Anwendungen, die als UTM-F-Anwendung generiert sind (APPLIMODE=FAST), werden Cluster-weit gültige Anwenderdaten auch gesichert. Dabei werden GSSB- und ULS-Daten genauso behandelt wie in UTM-Cluster-Anwendungen, die mit UTM-S generiert sind. Vorgangs-Daten von Benutzern mit RESTART=YES werden jedoch nur beim Abmelden des Benutzers anstatt bei jedem Transaktionsende geschrieben.

UTM-Generierung
UTM generation

Statische Konfiguration einer UTM-Anwendung mit dem UTM-Tool KDCDEF und Erzeugen des Anwendungsprogramms.

UTM-gesteuerte Queues
UTM controlled queue

Message Queues, bei denen der Abruf und die Weiterverarbeitung der Nachrichten vollständig durch openUTM gesteuert werden. Siehe auch Asynchron-Auftrag, Hintergrund-Auftrag und Asynchron-Nachricht.

UTM-S

Bei UTM-S-Anwendungen sichert openUTM neben den Verwaltungsdaten auch alle Benutzerdaten über ein Anwendungsende und einen Systemausfall hinaus. Außerdem garantiert UTM-S bei allen Störungen die Sicherheit und Konsistenz der Anwendungsdaten. Im Standardfall werden UTM-Anwendungen als UTM-S-Anwendungen (UTM-Secure) generiert.

UTM-SAT-Administration (BS2000-Systeme)
UTM SAT administration

Durch die UTM-SAT-Administration wird gesteuert, welche sicherheitsrelevanten UTM-Ereignisse, die im Betrieb der UTM-Anwendung auftreten, von SAT protokolliert werden sollen. Für die UTM-SAT-Administration wird eine besondere Berechtigung benötigt.

UTM-Seite
UTM page

Ist eine Speichereinheit, die entweder 2K, 4K oder 8K umfasst. In stand-alone UTM-Anwendungen kann die Größe einer UTM-Seite bei der Generierung der UTM-Anwendung auf 2K, 4K oder 8K gesetzt werden. In einer UTM-Cluster-Anwendung ist die Größe einer UTM-Seite immer 4K oder 8K. Pagepool und Wiederanlauf-Bereich der KDCFILE sowie UTM-Cluster-Dateien werden in Einheiten der Größe einer UTM-Seite unterteilt.

UTM Socket Protokoll (USP)
UTM socket protocol

Proprietäres Protokoll von openUTM oberhalb von TCP/IP zur Umsetzung der über die Socket-Schnittstelle empfangenen Bytestreams in Nachrichten.

UTM-System-Prozess
UTM system process

UTM-Prozess, der zusätzlich zu den per Startparameter angegebenen Prozessen gestartet wird und nur ausgewählte Aufträge bearbeitet. UTM-System-Prozesse dienen dazu, eine UTM-Anwendung auch bei sehr hoher Last reaktionsfähig zu halten.

UTM-Tool
UTM tool

Programm, das zusammen mit openUTM zur Verfügung gestellt und für bestimmte UTM-spezifische Aufgaben benötigt wird (z.B. zum Konfigurieren).

utmpfad (Unix-, Linux- und Windows-Systeme)
utmpath

Das Dateiverzeichnis unter dem die Komponenten von openUTM installiert sind, wird in diesem Handbuch als utmpfad bezeichnet.
Um einen korrekten Ablauf von openUTM zu garantieren, muss die Umgebungsvariable UTMPATH auf den Wert von utmpfad gesetzt werden. Auf Unix- und Linux-Systemen müssen Sie UTMPATH vor dem Starten einer UTM-Anwendung setzen. Auf Windows-Systemen wird UTMPATH passend zu der zuletzt installierten UTM-Version gesetzt.

Verarbeitungsschritt
processing step

Ein Verarbeitungsschritt beginnt mit dem Empfangen einer Dialog-Nachricht, die von einem Client oder einer anderen Server-Anwendung an die UTM-Anwendung gesendet wird. Der Verarbeitungsschritt endet entweder mit dem Senden einer Antwort und beendet damit auch den Dialog-Schritt oder er endet mit dem Senden einer Dialog-Nachricht an einen Dritten.

Verbindungs-Benutzerkennung
connection user ID

Benutzerkennung, unter der eine TS-Anwendung oder ein UPIC-Client direkt nach dem Verbindungsaufbau bei der UTM-Anwendung angemeldet wird. Abhängig von der Generierung des Clients (= LTERM-Partner) gilt:

- Die Verbindungs-Benutzerkennung ist gleich dem USER der LTERM-Anweisung (explizite Verbindungs-Benutzerkennung). Eine explizite Verbindungs-Benutzerkennung muss mit einer USER-Anweisung generiert sein und kann nicht als "echte" Benutzerkennung verwendet werden.
- Die Verbindungs-Benutzerkennung ist gleich dem LTERM-Partner (implizite Verbindungs-Benutzerkennung), wenn bei der LTERM-Anweisung kein USER angegeben wurde oder wenn ein LTERM-Pool generiert wurde.

In einer UTM-Cluster-Anwendung ist der Vorgang einer Verbindungs-Benutzerkennung (RESTART=YES bei LTERM oder USER) an die Verbindung gebunden und damit Knoten-lokal. Eine Verbindungs-Benutzerkennung, die mit RESTART=YES generiert ist, kann in jeder Knoten-Anwendung einen eigenen Vorgang haben.

Verbindungsbündel
connection bundle

siehe LTERM-Bündel.

Verschlüsselungsstufe
encryption level

Die Verschlüsselungsstufe legt fest, ob und inwieweit ein Client Nachrichten und Passwort verschlüsseln muss.

Verteilte Transaktion
distributed transaction

Transaktion, die sich über mehr als eine Anwendung erstreckt und in mehreren (Teil-)Transaktionen in verteilten Systemen ausgeführt wird.

Verteilte Transaktionsverarbeitung
Distributed Transaction Processing

Verteilte Verarbeitung mit verteilten Transaktionen.

Verteilte Verarbeitung
distributed processing

Bearbeitung von Dialog-Aufträgen durch mehrere Anwendungen oder Übermittlung von Hintergrundaufträgen an eine andere Anwendung. Für die verteilte Verarbeitung werden die höheren Kommunikationsprotokolle LU6.1 und OSI TP verwendet. Über openUTM-LU62 ist verteilte Verarbeitung auch mit LU6.2 Partnern möglich. Man unterscheidet verteilte Verarbeitung mit verteilten Transaktionen (Anwendungs-übergreifende Transaktionssicherung) und verteilte Verarbeitung ohne verteilte Transaktionen (nur lokale Transaktionssicherung). Die verteilte Verarbeitung wird auch Server-Server-Kommunikation genannt.

Vorgang (KDCS)
service

Ein Vorgang dient zur Bearbeitung eines Auftrags in einer UTM-Anwendung. Er setzt sich aus einer oder mehreren Transaktionen zusammen. Die erste Transaktion wird über den Vorgangs-TAC aufgerufen. Es gibt Dialog-Vorgänge und Asynchron-Vorgänge. openUTM stellt den Teilprogrammen eines Vorgangs gemeinsame Datenbereiche zur Verfügung. Anstelle des Begriffs Vorgang wird häufig auch der allgemeinere Begriff Service gebraucht.

Vorgangs-Kellerung (KDCS)
service stacking

Ein Terminal-Benutzer kann einen laufenden Dialog-Vorgang unterbrechen und einen neuen Dialog-Vorgang einschieben. Nach Beendigung des eingeschobenen Vorgangs wird der unterbrochene Vorgang fortgesetzt.

Vorgangs-Kettung (KDCS)
service chaining

Bei Vorgangs-Kettung wird nach Beendigung eines Dialog-Vorgangs ohne Angabe einer Dialog-Nachricht ein Folgevorgang gestartet.

Vorgangs-TAC (KDCS)
service TAC

Transaktionscode, mit dem ein Vorgang gestartet wird.

Vorgangs-Wiederanlauf (KDCS)
service restart

Wird ein Vorgang unterbrochen, z.B. infolge Abmeldens des Terminal-Benutzers oder Beendigung der UTM-Anwendung, führt openUTM einen Vorgangs-Wiederanlauf durch. Ein Asynchron-Vorgang wird neu gestartet oder beim zuletzt erreichten Sicherungspunkt fortgesetzt, ein Dialog-Vorgang wird beim zuletzt erreichten Sicherungspunkt fortgesetzt. Für den Terminal-Benutzer wird der Vorgangs-Wiederanlauf eines Dialog-Vorgangs als Bildschirm-Wiederanlauf sichtbar, sofern am letzten Sicherungspunkt eine Dialog-Nachricht an den Terminal-Benutzer gesendet wurde.

Warmstart
warm start

Start einer UTM-S-Anwendung nach einer vorhergehenden abnormalen Beendigung. Dabei wird die Anwendungsinformation auf den zuletzt erreichten konsistenten Zustand gesetzt. Unterbrochene Dialog-Vorgänge werden dabei auf den zuletzt erreichten Sicherungspunkt zurückgesetzt, so dass die Verarbeitung an dieser Stelle wieder konsistent aufgenommen werden kann (Vorgangs-Wiederanlauf). Unterbrochene Asynchron-Vorgänge werden zurückgesetzt und neu gestartet oder beim zuletzt erreichten Sicherungspunkt fortgesetzt.
Bei UTM-F-Anwendungen werden beim Start nach einer vorhergehenden abnormalen Beendigung lediglich die dynamisch geänderten Konfigurationsdaten auf den zuletzt erreichten konsistenten Zustand gesetzt.
In UTM-Cluster-Anwendungen werden die globalen Sperren auf GSSB und ULS, die bei der abnormalen Beendigung von dieser Knoten-Anwendung gehalten wurden, aufgehoben. Außerdem werden Benutzer, die zum Zeitpunkt der abnormalen Beendigung an dieser Knoten-Anwendung angemeldet waren, abgemeldet.

Web Service
web service

Anwendung, die auf einem Web-Server läuft und über eine standardisierte und programmatische Schnittstelle (öffentlich) verfügbar ist. Die Web Services-Technologie ermöglicht es, UTM-Teilprogramme für moderne Web-Client-Anwendungen verfügbar zu machen, unabhängig davon, in welcher Programmiersprache sie entwickelt wurden.

WebAdmin
WebAdmin

Web-basiertes Tool zur Administration von openUTM-Anwendungen über Web-Browser. WebAdmin enthält neben dem kompletten Funktionsumfang der Programmschnittstelle zur Administration noch zusätzliche Funktionen.

Wiederanlauf
restart

siehe Bildschirm-Wiederanlauf,
siehe Vorgangs-Wiederanlauf.

WinAdmin
WinAdmin

Java-basiertes Tool zur Administration von openUTM-Anwendungen über eine grafische Oberfläche. WinAdmin enthält neben dem kompletten Funktionsumfang der Programmschnittstelle zur Administration noch zusätzliche Funktionen.

Workload Capture & Replay
workload capture & replay

Programmfamilie zur Simulation von Lastsituationen, bestehend aus den Haupt-Komponenten UPIC Capture, UPIC Analyzer und Upic Replay und auf Unix-, Linux- und Windows-Systemen dem Dienstprogramm kdcsort. Mit Workload Capture & Replay lassen sich UPIC-Sessions mit UTM-Anwendungen aufzeichnen, analysieren und mit veränderten Lastparametern wieder abspielen.

Workprozess (Unix-, Linux- und Windows-Systeme)
work process

Prozess, in dem die Services der UTM-Anwendung ablaufen.

WS4UTM

WS4UTM (**W**eb**S**ervices for open**UTM**) ermöglicht es Ihnen, auf komfortable Weise einen Service einer UTM-Anwendung als Web Service zur Verfügung zu stellen.

XATMI

XATMI (X/Open Application Transaction Manager Interface) ist eine von X/Open standardisierte Programmschnittstelle für die Programm-Programm-Kommunikation in offenen Netzen.
Das in openUTM implementierte XATMI genügt der XATMI CAE Specification von X/Open. Die Schnittstelle steht in COBOL und C zur Verfügung. XATMI in openUTM kann über die Protokolle OSI TP, LU6.1 und UPIC kommunizieren.

XHCS (BS2000-Systeme)

XHCS (Extended Host Code Support) ist ein BS2000-Softwareprodukt für die Unterstützung internationaler Zeichensätze.

XML

XML (eXtensible Markup Language) ist eine vom W3C (WWW-Konsortium) genormte Metasprache, in der Austauschformate für Daten und zugehörige Informationen definiert werden können.

Zeitgesteuerter Auftrag
time-driven job

Auftrag, der von openUTM bis zu einem definierten Zeitpunkt in einer Message Queue zwischengespeichert und dann an den Empfänger weitergeleitet wird. Empfänger kann sein: ein Asynchron-Vorgang der selben Anwendung, eine TAC-Queue, eine Partner-Anwendung, ein Terminal oder ein Drucker. Zeitgesteuerte Aufträge können nur von KDCS-Teilprogrammen erteilt werden.

Zugangskontrolle
system access control

Prüfung durch openUTM, ob eine bestimmte Benutzerkennung berechtigt ist, mit der UTM-Anwendung zu arbeiten. Die Berechtigungsprüfung entfällt, wenn die UTM-Anwendung ohne Benutzerkennungen generiert wurde.

Zugriffskontrolle
data access control

Prüfung durch openUTM, ob der Kommunikationspartner berechtigt ist, auf ein bestimmtes Objekt der Anwendung zuzugreifen. Die Zugriffsrechte werden als Bestandteil der Konfiguration festgelegt.

Zugriffspunkt
access point

siehe Dienstzugriffspunkt.

16 Abkürzungen

ACSE	Association Control Service Element
AEQ	Application Entity Qualifier
AES	Advanced Encryption Standard
AET	Application Entity Title
APT	Application Process Title
ASCII	American Standard Code for Information Interchange
ASE	Application Service Element
Axis	Apache eXtensible Interaction System
BCAM	Basic Communication Access Method
BER	Basic Encoding Rules
BLS	Binder-Lader-Starter (BS2000-Systeme)
CCP	Communication Control Program
CCR	Commitment, Concurrency and Recovery
CCS	Codierter Zeichensatz (Coded Character Set)
CCSN	Name des codierten Zeichensatzes (Coded Character Set Name)
CICS	Customer Information Control System (IBM)
CID	Control Identification
CMX	Communication Manager in Unix-, Linux- und Windows-Systemen
COM	Component Object Model
CPI-C	Common Programming Interface for Communication
CRM	Communication Resource Manager
CRTE	Common Runtime Environment (BS2000-Systeme)
DB	Database
DBH	Database Handler
DC	Data Communication
DCAM	Data Communication Access Method

DES	Data Encryption Standard
DLS	Distributed Lock Manager (BS2000-Systeme)
DMS	Data Management System
DNS	Domain Name Service
DSS	Datensichtstation (=Terminal)
DTD	Document Type Definition
DTP	Distributed Transaction Processing
DVS	Datenverwaltungssystem
EBCDIC	Extended Binary-Coded Decimal Interchange Code
EJB	Enterprise JavaBeans™
FGG	File Generation Group
FHS	Format Handling System
FT	File Transfer
GCM	Galois/Counter Mode
GSSB	Globaler Sekundärer Speicherbereich
HIPLEX®	Highly Integrated System Complex (BS2000-Systeme)
HLL	High-Level Language
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IFG	Interaktiver Format-Generator
ILCS	Inter Language Communication Services (BS2000-Systeme)
IMS	Information Management System (IBM)
IPC	Inter-Process-Communication
IRV	Internationale Referenzversion
ISO	International Organization for Standardization
Java EE	Java Platform, Enterprise Edition
JCA	Java EE Connector Architecture
JDK	Java Development Kit

KAA	KDCS Application Area
KB	Kommunikationsbereich
KBPROG	KB-Programmbereich
KDCADMI	KDC Administration Interface
KDCS	Kompatible Datenkommunikationsschnittstelle
KTA	KDCS Task Area
LAN	Local Area Network
LCF	Local Configuration File
LLM	Link and Load Module (BS2000-Systeme)
LSSB	Lokaler Sekundärer Speicherbereich
LU	Logical Unit
MQ	Message Queuing
MSCF	Multiple System Control Facility (BS2000-Systeme)
NB	Nachrichtenbereich
NEA	Netzwerkarchitektur bei BS2000-Systemen
NFS	Network File System/Service
NLS	Unterstützung der Landessprache (Native Language Support)
OLTP	Online Transaction Processing
OML	Object Modul Library
OSI	Open System Interconnection
OSI TP	Open System Interconnection Transaction Processing
OSS	OSI Session Service
PCMX	Portable Communication Manager
PID	Prozess-Identifikation
PIN	Persönliche Identifikationsnummer
PLU	Primary Logical Unit
PTC	Prepare to commit
RAV	Rechenzentrums-Abrechnungs-Verfahren

RDF	Resource Definition File
RM	Resource Manager
RSA	Encryption-Algorithmus nach Rivest, Shamir, Adleman
RSO	Remote SPOOL Output (BS2000-Systeme)
RTS	Runtime System (Laufzeitsystem)
SAT	Security Audit Trail (BS2000-Systeme)
SECOS	Security Control System
SEM	SE Manager
SGML	Standard Generalized Markup Language
SLU	Secondary Logical Unit
SM2	Software Monitor 2
SNA	Systems Network Architecture
SOA	Service-oriented Architecture
SOAP	Simple Object Access Protocol
SPAB	Standard Primärer Arbeitsbereich
SQL	Structured Query Language
SSB	Sekundärer Speicherbereich
SSL	Secure Socket Layer
SSO	Single-Sign-On
TAC	Transaktionscode
TCEP	Transport Connection End Point
TCP/IP	Transport Control Protocol / Internet Protocol
TIAM	Terminal Interactive Access Method
TLS	Terminal-spezifischer Langzeitspeicher
TLS	Transport Layer Security
TM	Transaction Manager
TNS	Transport Name Service
TP	Transaction Processing (Transaktions-Betrieb)
TPR	Task privileged (privilegierter Funktionszustand des BS2000-Systems)

TPSU	Transaction Protocol Service User
TSAP	Transport Service Access Point
TSN	Task Sequence Number
TU	Task user (nicht privilegierter Funktionszustand des BS2000-Systems)
TX	Transaction Demarcation (X/Open)
UDDI	Universal Description, Discovery and Integration
UDS	Universelles Datenbanksystem
UDT	Unstructured Data Transfer
ULS	User-spezifischer Langzeitspeicher
UPIC	Universal Programming Interface for Communication
USP	UTM-Socket-Protokoll
UTM	Universeller Transaktionsmonitor
UTM-D	UTM-Funktionen für verteilte Verarbeitung („Distributed“)
UTM-F	Schnelle UTM-Variante („Fast“)
UTM-S	UTM-Sicherheitsvariante
UTM-XML	XML-Schnittstelle von openUTM
VGID	Vorgangs-Identifikation
VTSU	Virtual Terminal Support
VTV	Verteilte Transaktionsverarbeitung
VV	Verteilte Verarbeitung
WAN	Wide Area Network
WS4UTM	WebServices for openUTM
WSDD	Web Service Deployment Descriptor
WSDL	Web Services Description Language
XA	X/Open Access Interface (Schnittstelle von X/Open zum Zugriff auf Resource Manager)
XAP	X/OPEN ACSE/Presentation programming interface
XAP-TP	X/OPEN ACSE/Presentation programming interface Transaction Processing extension
XATMI	X/Open Application Transaction Manager Interface
XCS	Cross Coupled System

XHCS	eXtended Host Code Support
XML	eXtensible Markup Language

17 Literatur

Die Handbücher finden Sie im Internet unter https://bs2manuals.ts.fujitsu.com.

Dokumentation zu openUTM

openUTM
Konzepte und Funktionen
Benutzerhandbuch

openUTM
Anwendungen programmieren mit KDCS für COBOL, C und C++
Basishandbuch

openUTM
Anwendungen generieren
Benutzerhandbuch

openUTM
Einsatz von UTM-Anwendungen auf BS2000-Systemen
Benutzerhandbuch

openUTM
Einsatz von UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen
Benutzerhandbuch

openUTM
Anwendungen administrieren
Benutzerhandbuch

openUTM
Meldungen, Test und Diagnose auf BS2000-Systemen
Benutzerhandbuch

openUTM
Meldungen, Test und Diagnose auf Unix-, Linux- und Windows-Systemen
Benutzerhandbuch

openUTM
Anwendungen erstellen mit X/Open-Schnittstellen
Benutzerhandbuch

openUTM
XML für openUTM

openUTM-Client (Unix-Systeme) **für Trägersystem OpenCPIC**
Client-Server-Kommunikation mit openUTM
Benutzerhandbuch

openUTM-Client für Trägersystem UPIC
Client-Server-Kommunikation mit openUTM
Benutzerhandbuch

openUTM WinAdmin
Grafischer Administrationsarbeitsplatz für openUTM
Beschreibung und Online-Hilfe

openUTM WebAdmin
Web-Oberfläche zur Administration von openUTM
Beschreibung und Online-Hilfe

openUTM, openUTM-LU62
Verteilte Transaktionsverarbeitung
zwischen openUTM und CICS-, IMS- und LU6.2-Anwendungen
Benutzerhandbuch

openUTM (BS2000)
Anwendungen programmieren mit KDCS für Assembler
Ergänzung zum Basishandbuch

openUTM (BS2000)
Anwendungen programmieren mit KDCS für Fortran
Ergänzung zum Basishandbuch

openUTM (BS2000)
Anwendungen programmieren mit KDCS für Pascal-XT
Ergänzung zum Basishandbuch

openUTM (BS2000)
Anwendungen programmieren mit KDCS für PL/I
Ergänzung zum Basishandbuch

WS4UTM (Unix- und Windows-Systeme**)**
Web-Services für openUTM

Dokumentation zum openSEAS-Produktumfeld

BeanConnect
Benutzerhandbuch

openUTM-JConnect
Verbindung von Java-Clients zu openUTM
Benutzerdokumentation und Java-Docs

WebTransactions
Konzepte und Funktionen

WebTransactions
Template-Sprache

WebTransactions
Anschluss an openUTM-Anwendungen über UPIC

WebTransactions
Anschluss an MVS-Anwendungen

WebTransactions
Anschluss an OSD-Anwendungen

Dokumentation zum BS2000-Umfeld

AID Advanced Interactive Debugger
Basishandbuch
Benutzerhandbuch

AID Advanced Interactive Debugger
Testen von COBOL-Programmen
Benutzerhandbuch

AID Advanced Interactive Debugger
Testen von C/C++-Programmen
Benutzerhandbuch

BCAM
BCAM Band 1/2
Benutzerhandbuch

BINDER
Benutzerhandbuch

BS2000 OSD/BC
Kommandos Band 1-7
Benutzerhandbuch

BS2000 OSD/BC
Makroaufrufe an den Ablaufteil
Benutzerhandbuch

BS2IDE
Eclipse-based Integrated Development Environment for BS2000
User Guide and Installation Guide
Webseite: *https://bs2000.ts.fujitsu.com/bs2ide/*

BLSSERV
Bindelader-Starter in BS2000/OSD
Benutzerhandbuch

DCAM
COBOL-Aufrufe
Benutzerhandbuch

DCAM
Makroaufrufe
Benutzerhandbuch

DCAM
Programmschnittstellen
Beschreibung

FHS
Formatierungssystem für openUTM, TIAM, DCAM
Benutzerhandbuch

IFG für FHS
Benutzerhandbuch

HIPLEX AF
Hochverfügbarkeit von Anwendungen in BS2000/OSD
Produkthandbuch

HIPLEX MSCF
BS2000-Rechner im Verbund
Benutzerhandbuch

IMON
Installationsmonitor
Benutzerhandbuch

LMS
SDF-Format
Benutzerhandbuch

MT9750 (MS Windows)
9750-Emulation unter Windows
Produkthandbuch

OMNIS/OMNIS-MENU
Funktionen und Kommandos
Benutzerhandbuch

OMNIS/OMNIS-MENU
Administration und Programmierung
Benutzerhandbuch

OSS (BS2000)
OSI Session Service
User Guide

openSM2
Software Monitor
Benutzerhandbuch

RSO
Remote SPOOL Output
Benutzerhandbuch

SECOS
Security Control System
Benutzerhandbuch

SECOS
Security Control System
Tabellenheft

SESAM/SQL
Datenbankbetrieb
Benutzerhandbuch

TIAM
Benutzerhandbuch

UDS/SQL
Datenbankbetrieb
Benutzerhandbuch

Unicode im BS2000/OSD
Übersichtshandbuch

VTSU
Virtual Terminal Support
Benutzerhandbuch

XHCS
8-bit-Code- und Unicode-Unterstützung im BS2000/OSD
Benutzerhandbuch

Dokumentation zum Umfeld von Unix-, Linux- und Windows-Systemen

CMX V6.0 (Unix-Systeme)
Betrieb und Administration
Benutzerhandbuch

CMX V6.0
CMX-Anwendungen programmieren
Programmierhandbuch

OSS (UNIX)
OSI Session Service
User Guide

PRIMECLUSTER™
Konzept (Solaris, Linux)
Benutzerhandbuch

openSM2
Die Dokumentation zu openSM2 wird in Form von ausführlichen Online-Hilfen bereitgestellt, die mit dem Produkt ausgeliefert werden.

Sonstige Literatur

CPI-C
X/Open CAE Specification
Distributed Transaction Processing:
The CPI-C Specification, Version 2
ISBN 1 85912 135 7

Reference Model
X/Open Guide
Distributed Transaction Processing:
Reference Model, Version 2
ISBN 1 85912 019 9

REST
Architectural Styles and the Design of Network-based Software Architectures
Dissertation Roy Fielding

TX
X/Open CAE Specification
Distributed Transaction Processing:
The TX (Transaction Demarcation) Specification
ISBN 1 85912 094 6

XATMI
X/Open CAE Secification
Distributed Transaction Processing
The XATMI Specification
ISBN 1 85912 130 6

XML
Spezifikation des W3C (www – Konsortium)
Webseite: *http://www.w3.org/XML*